http://tinyurl.org/ift3355

IFT 3355: INFOGRAPHIE

REVUE

FİN

LES COMPOSITIONS SIMPLES

$$Tr(x_1, y_1, z_1) \cdot Tr(x_2, y_2, z_2) = ?$$

$$Tr(x_2, y_2, z_2) \cdot Tr(x_1, y_1, z_1) \ ? \ Tr(x_2, y_2, z_2) \cdot Tr(x_1, y_1, z_1)$$

$$Scale(a, b, c) \cdot Scale(d, e, f) = ?$$

$$Scale(a, b, c) \cdot Scale(d, e, f) \ ? \ Scale(d, e, f) \cdot Scale(a, b, c)$$

$$Rot(\alpha, 0,0,1) \cdot Rot(\beta, 0,0,1) = ?$$

$$Rot(\alpha, 0,0,1) \cdot Rot(\beta, 0,0,1) ? Rot(\beta, 0,0,1) \cdot Rot(\alpha, 0,0,1)$$

LES COMPOSITIONS SIMPLES

$$Tr(x_1, y_1, z_1) \cdot Tr(x_2, y_2, z_2) = Tr(x_1 + x_2, y_1 + y_2, z_1 + z_2)$$
$$Tr(x_2, y_2, z_2) \cdot Tr(x_1, y_1, z_1) = Tr(x_2, y_2, z_2) \cdot Tr(x_1, y_1, z_1)$$

$$Scale(a, b, c) \cdot Scale(d, e, f) = Scale(ad, be, cf)$$
$$Scale(a, b, c) \cdot Scale(d, e, f) = Scale(d, e, f) \cdot Scale(a, b, c)$$

$$Rot(\alpha, 0,0,1) \cdot Rot(\beta, 0,0,1) = Rot(\alpha + \beta, 0,0,1)$$
$$Rot(\alpha, 0,0,1) \cdot Rot(\beta, 0,0,1) = Rot(\beta, 0,0,1) \cdot Rot(\alpha, 0,0,1)$$

LES COMPOSITIONS PLUS COMPLEXES

$$Tr(x, y, z) \cdot Scale(a, b, c) \; ? \; Scale(a, b, c) \cdot Tr(x, y, z)$$

LES COMPOSITIONS PLUS COMPLEXES

$$Tr(x, y, z) \cdot Scale(a, b, c) \neq Scale(a, b, c) \cdot Tr(x, y, z)$$

$$Tr(x, y, z) \cdot Scale(a, b, c) = Scale(a, b, c) \cdot Tr(\frac{x}{a}, \frac{y}{b}, \frac{z}{c})$$

LES COMPOSITIONS PLUS COMPLEXES

$$Tr(x,y,z) \cdot Scale(a,b,c) \neq Scale(a,b,c) \cdot Tr(x,y,z)$$

$$Tr(x,y,z) \cdot Scale(a,b,c) = Scale(a,b,c) \cdot Tr\left(\frac{x}{a},\frac{y}{b},\frac{z}{c}\right)$$

$$Tr(x,y,z) \cdot Rot(\alpha,0,0,1) \neq Rot(\alpha,0,0,1) \cdot Tr(x,y,z)$$

$$Rot(\alpha,0,0,1) \cdot Rot(\beta,0,1,0) \; ? \; Rot(\beta,0,1,0) \cdot Rot(\alpha,0,0,1)$$

LES COMPOSITIONS PLUS COMPLEXES

$$Tr(x,y,z) \cdot Scale(a,b,c) \neq Scale(a,b,c) \cdot Tr(x,y,z)$$

$$Tr(x,y,z) \cdot Scale(a,b,c) = Scale(a,b,c) \cdot Tr\left(\frac{x}{a},\frac{y}{b},\frac{z}{c}\right)$$

$$Tr(x,y,z) \cdot Rot(\alpha,0,0,1) \neq Rot(\alpha,0,0,1) \cdot Tr(x,y,z)$$

$$Rot(\alpha,0,0,1) \cdot Rot(\beta,0,1,0) \neq Rot(\beta,0,1,0) \cdot Rot(\alpha,0,0,1)$$

LES COMPOSITIONS PLUS COMPLEXES

$$Tr(x,y,z) \cdot Scale(a,b,c) \neq Scale(a,b,c) \cdot Tr(x,y,z)$$
$$Tr(x,y,z) \cdot Scale(a,b,c) = Scale(a,b,c) \cdot Tr\left(\frac{x}{a},\frac{y}{b},\frac{z}{c}\right)$$

$$Tr(x,y,z) \cdot Rot(\alpha,0,0,1) \neq Rot(\alpha,0,0,1) \cdot Tr(x,y,z)$$
$$Rot(\alpha,0,0,1) \cdot Rot(\beta,0,1,0) \neq Rot(\beta,0,1,0) \cdot Rot(\alpha,0,0,1)$$
$$Scale(a,a,a) \cdot Rot(\beta,0,0,1) \,?\, Rot(\beta,0,0,1) \cdot Scale(a,a,a)$$

LES COMPOSITIONS PLUS COMPLEXES

$$Tr(x,y,z) \cdot Scale(a,b,c) \neq Scale(a,b,c) \cdot Tr(x,y,z)$$
$$Tr(x,y,z) \cdot Scale(a,b,c) = Scale(a,b,c) \cdot Tr\left(\frac{x}{a},\frac{y}{b},\frac{z}{c}\right)$$

$$Tr(x,y,z) \cdot Rot(\alpha,0,0,1) \neq Rot(\alpha,0,0,1) \cdot Tr(x,y,z)$$
$$Rot(\alpha,0,0,1) \cdot Rot(\beta,0,1,0) \neq Rot(\beta,0,1,0) \cdot Rot(\alpha,0,0,1)$$
$$Scale(a,a,a) \cdot Rot(\beta,0,0,1) = Rot(\beta,0,0,1) \cdot Scale(a,a,a)$$

LES COMPOSITIONS PLUS COMPLEXES

$$Tr(x,y,z) \cdot Scale(a,b,c) \neq Scale(a,b,c) \cdot Tr(x,y,z)$$
$$Tr(x,y,z) \cdot Scale(a,b,c) = Scale(a,b,c) \cdot Tr\left(\frac{x}{a},\frac{y}{b},\frac{z}{c}\right)$$

$$Tr(x,y,z) \cdot Rot(\alpha,0,0,1) \neq Rot(\alpha,0,0,1) \cdot Tr(x,y,z)$$
$$Rot(\alpha,0,0,1) \cdot Rot(\beta,0,1,0) \neq Rot(\beta,0,1,0) \cdot Rot(\alpha,0,0,1)$$
$$Scale(a,a,a) \cdot Rot(\beta,0,0,1) = Rot(\beta,0,0,1) \cdot Scale(a,a,a)$$

$$Scale(a,b,c) \cdot Rot(\beta,0,0,1) \neq Rot(\beta,0,0,1) \cdot Scale(a,b,c)$$

LE CHANGEMENT D'ÉCHELLE NON UNIFORME **APRÈS** LA ROTATION

- OK!
- $M = Rot(\beta, 0,0,1) \cdot Scale(a, b, c) =$

$$\begin{pmatrix} \cos(\beta) & -\sin(\beta) & 0 & 0 \\ \sin(\beta) & \cos(\beta) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} a & & & \\ & b & & \\ & & c & \\ & & & 1 \end{pmatrix} =$$

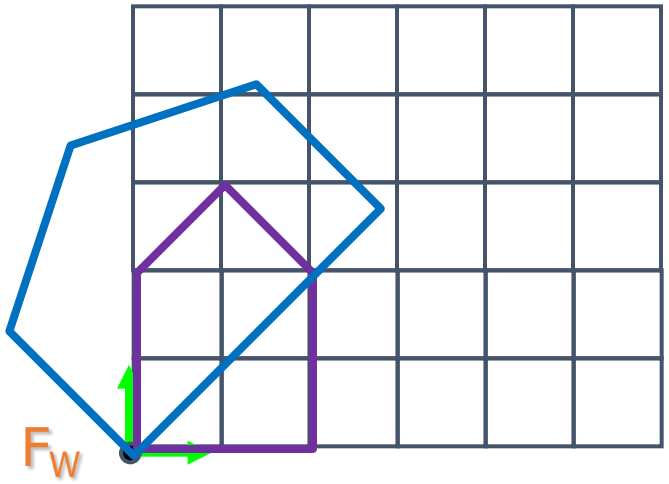
$$\begin{pmatrix} \mathrm{a} \cdot \cos(\beta) & -b \cdot \sin(\beta) & 0 & 0 \\ \mathrm{a} \cdot \sin(\beta) & b \cdot \cos(\beta) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

LE CHANGEMENT D'ÉCHELLE NON UNIFORME **APRÈS** LA ROTATION

- OK!
- $M = Rot(\beta, 0,0,1) \cdot Scale(a, b, c) =$

$$\begin{pmatrix} \mathrm{a} \cdot \cos(\beta) & -b \cdot \sin(\beta) & 0 & 0 \\ \mathrm{a} \cdot \sin(\beta) & b \cdot \cos(\beta) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

- Les nouveaux vecteurs de la base sont orthogonaux!
- La rotation + le changement d'échelle, comme on s'y attendrait


F_w

LE CHANGEMENT D'ÉCHELLE NON UNIFORME **AVANT** LA ROTATION

- Pas quelque chose qu'on s'attend!
- $M = Scale(a, b, c) \cdot Rot(\beta, 0,0,1) =$

$$\begin{pmatrix} a & & & \\ & b & & \\ & & c & \\ & & & 1 \end{pmatrix} \cdot \begin{pmatrix} \cos(\beta) & -\sin(\beta) & 0 & 0 \\ \sin(\beta) & \cos(\beta) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} =$$

$$\begin{pmatrix} \mathrm{a} \cdot \cos(\beta) & -a \cdot \sin(\beta) & 0 & 0 \\ \mathrm{b} \cdot \sin(\beta) & b \cdot \cos(\beta) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

LE CHANGEMENT D'ÉCHELLE NON UNIFORME **AVANT** LA ROTATION

- Pas quelque chose qu'on s'attend!
- $M = Scale(a, b, c) \cdot Rot(\beta, 0,0,1) =$

$$\begin{pmatrix} \mathrm{a} \cdot \cos(\beta) & -a \cdot \sin(\beta) & 0 & 0 \\ \mathrm{b} \cdot \sin(\beta) & b \cdot \cos(\beta) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

- Les vecteurs de la base ne sont pas orthogonaux!

LES TRANSFORMATIONS INVERSES

$$Tr(x, y, z)^{-1} = ?$$

$$Rot(\alpha, 0,0,1)^{-1} = ?$$

$$Scale(a, b, c)^{-1} = ?$$

LES TRANSFORMATIONS INVERSES

$$Tr(x,y,z)^{-1} = Tr(-x,-y,-z)$$

$$Rot(\alpha,0,0,1)^{-1} = Rot(-\alpha,0,0,1) = Rot(\alpha,0,0,1)^{T}$$ (orthogonal!)

$$Scale(a,b,c)^{-1} = Scale\left(\frac{1}{a},\frac{1}{b},\frac{1}{c}\right)$$

LES TRANSFORMATIONS INVERSES

$$M = Tr(1,2,3) \cdot Rot(45,0,0,1) \cdot Scale(2,2,2)$$

$$M^{-1} = ?$$

LES TRANSFORMATIONS INVERSES

$$M = Tr(1,2,3) \cdot Rot(45,0,0,1) \cdot Scale(2,2,2)$$

$$M^{-1} = \big(Scale(2,2,2)\big)^{-1} \cdot \big(Rot(45,0,0,1)\big)^{-1} \cdot \big(Tr(1,2,3)\big)^{-1}$$

$$M^{-1} = Scale(0.5,0.5,0.5) \cdot Rot(-45,0,0,1) \cdot Tr(-1,-2,-3)$$

LES MATHS

LES PRODUITS

- Scalaire

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} \bullet \begin{bmatrix} a \\ b \\ c \end{bmatrix} = x * a + y * b + z * c$$

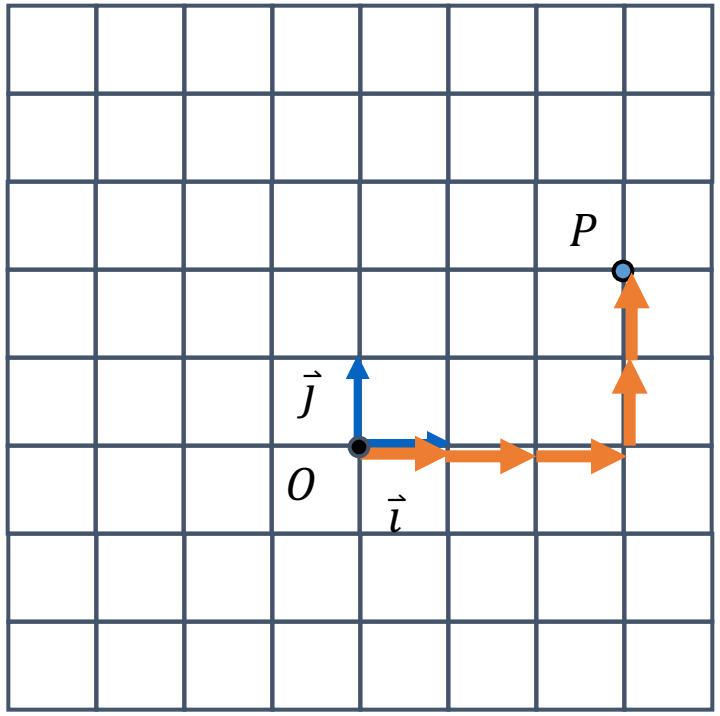
- Utilisé pour?
- Vectoriel

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} \times \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} = \begin{bmatrix} u_2 v_3 - u_3 v_2 \\ u_3 v_1 - u_1 v_3 \\ u_1 v_2 - u_2 v_1 \end{bmatrix}$$

- Utilisé pour?

LES SYSTÈMES DE COORDONNÉES

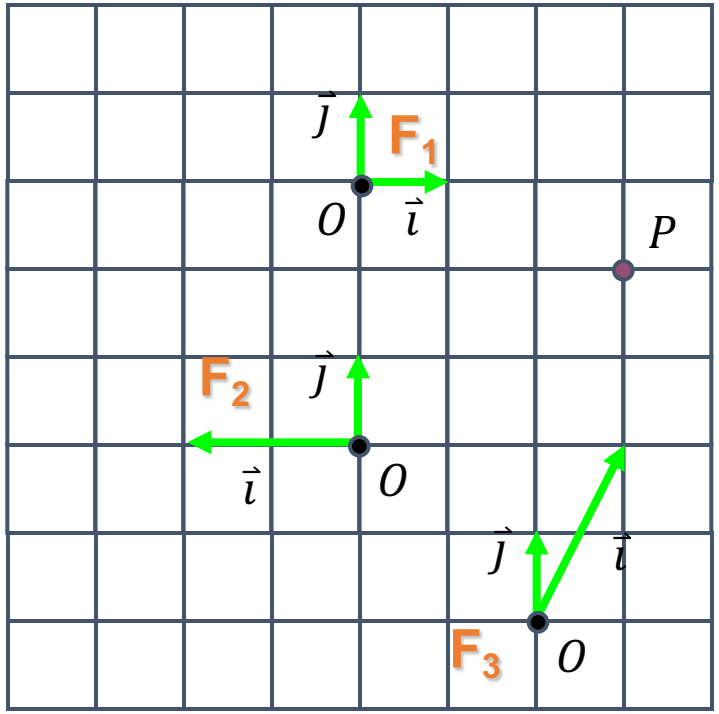
Un système de coordonnées = origine + vecteurs de base



$$P = O + x\vec{\imath} + y\vec{j}$$

LES MATHS

- Travailler avec des systèmes d'axe



$$P = O + x\vec{i} + y\vec{j}$$

F_1 P(3,-1)

F_2 P(-1.5,2)

F_3 P(1,2)

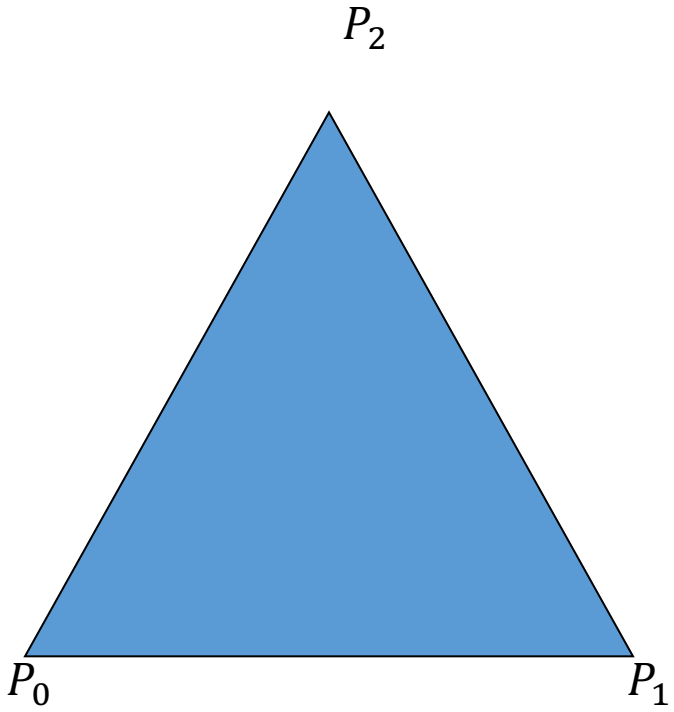
LE TRIANGLE

- La normale

$$n = \frac{(P_1 - P_0) \times (P_2 - P_0)}{\|(P_1 - P_0) \times (P_2 - P_0)\|}$$

- L'aire

$$A = \frac{1}{2} \left\| \overrightarrow{P_0 P_1} \times \overrightarrow{P_0 P_2} \right\|$$

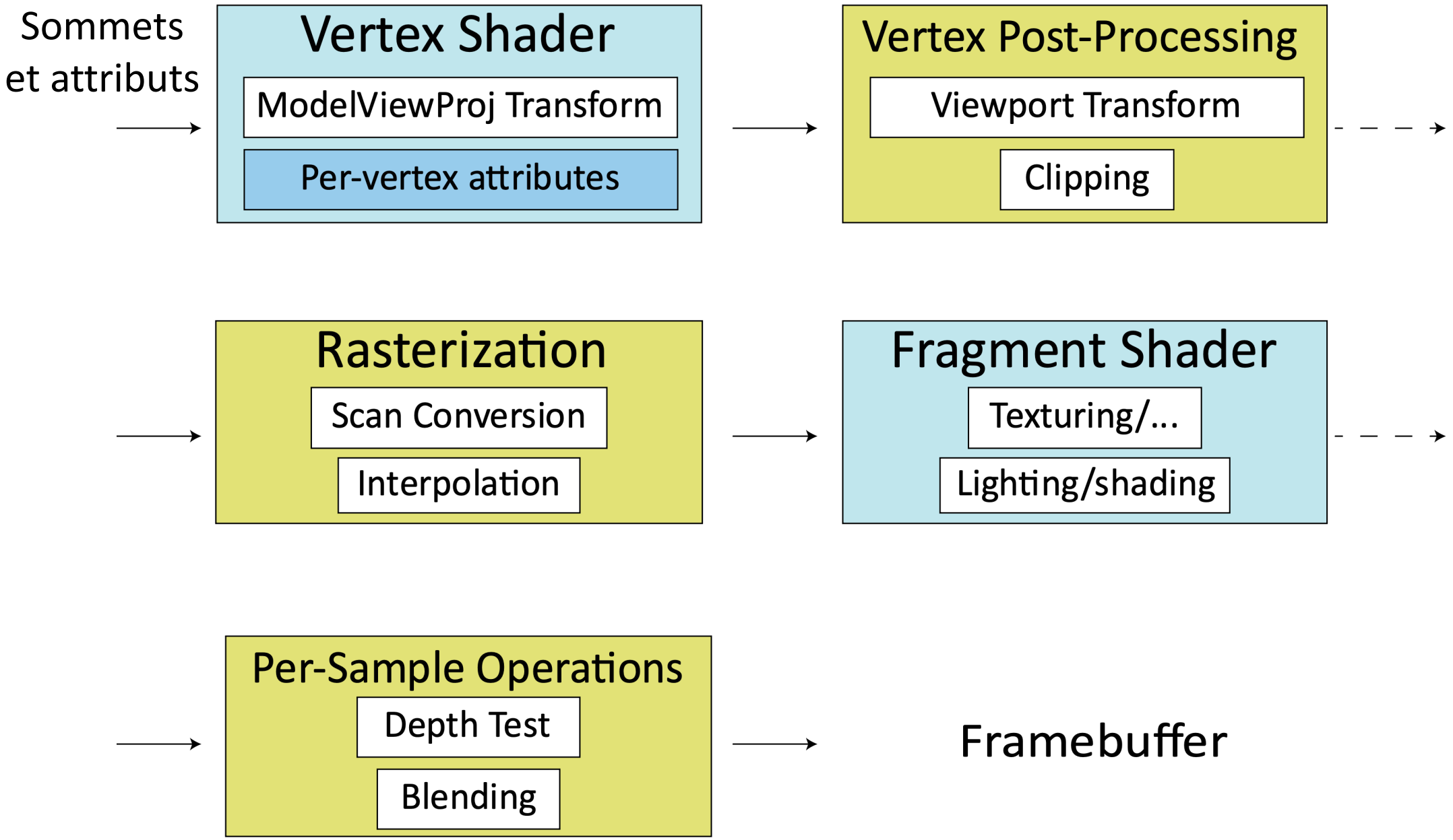


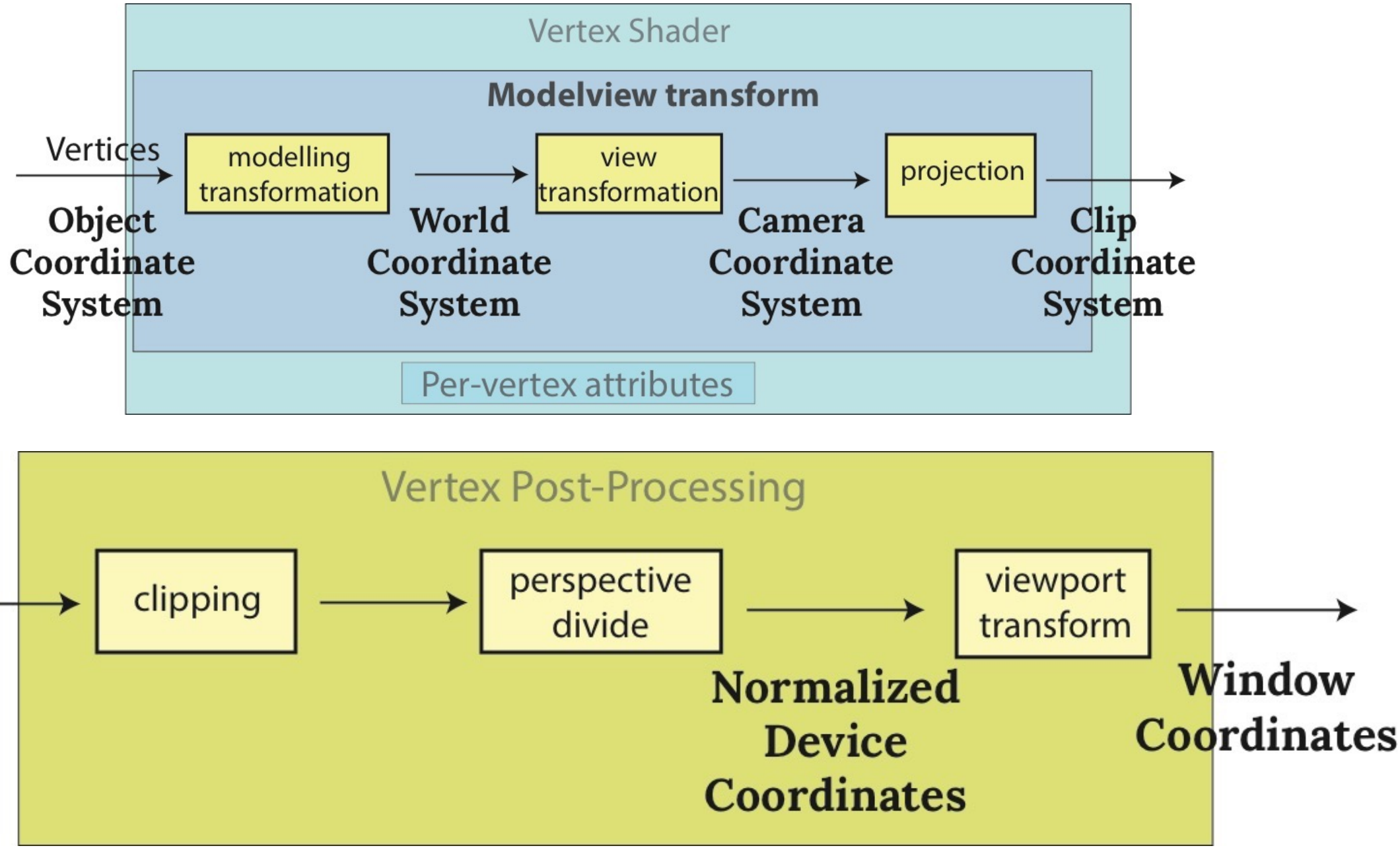
LE PLAN

- L'équation implicite: $Ax + By + Cz + D = 0$
  - Normaliser (une option): $A^2 + B^2 + C^2 = 1$
  - (A, B, C) - normale au plan

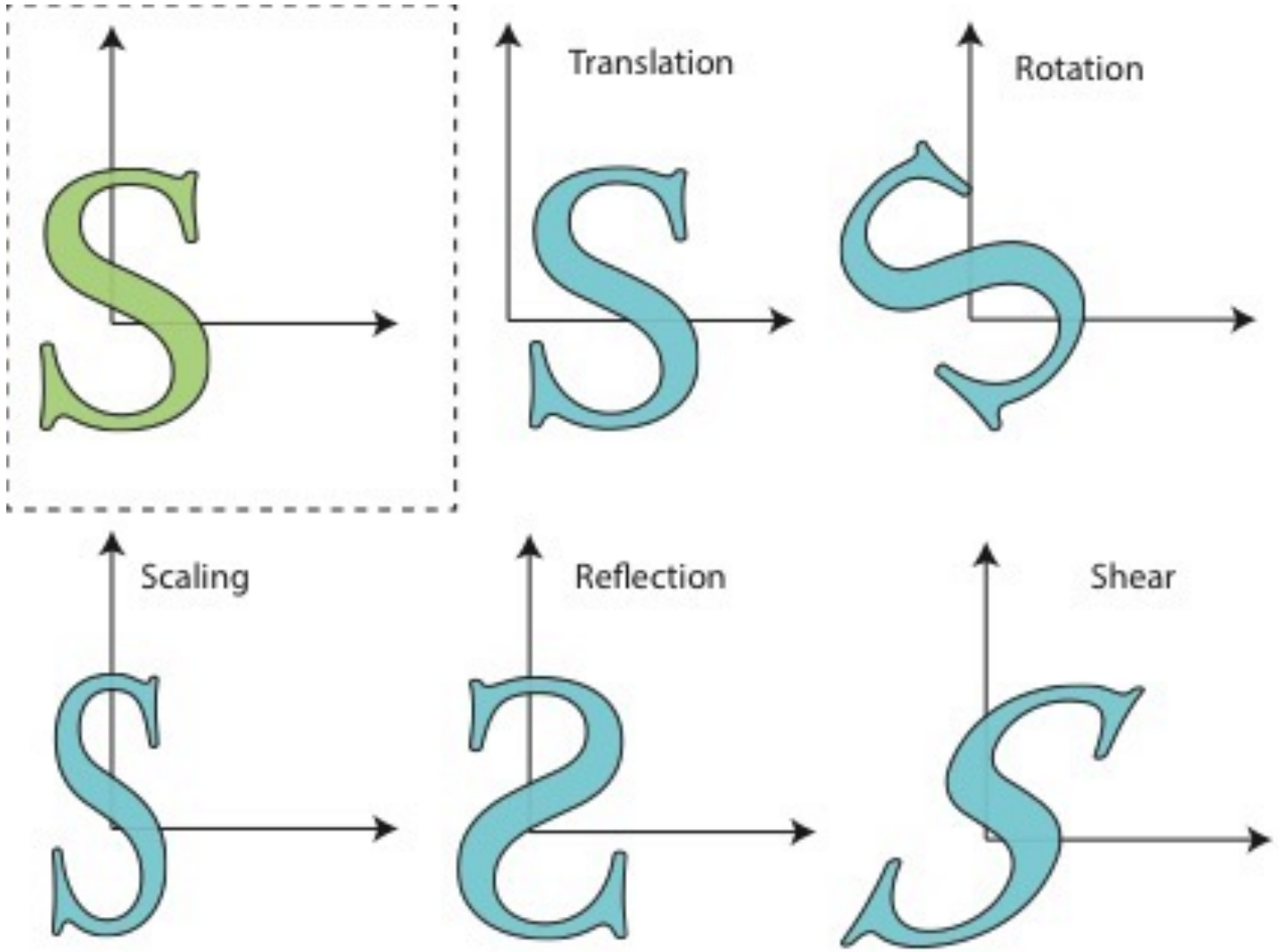
- Comment trouver l'équation si on a trois points?

SE SOUVENIR DU PIPELINE




Vertex Shader
Modelview transform
Vertices
modelling transformation
view transformation
projection
Object Coordinate System
World Coordinate System
Camera Coordinate System
Clip Coordinate System
Per-vertex attributes
Vertex Post-Processing
clipping
perspective divide
viewport transform
Normalized Device Coordinates
Window Coordinates

LES TRANSFORMATIONS AFFINES



LES COORDONNÉES HOMOGÈNES

Les coordonnées homogènes $\begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} \rightarrow \begin{bmatrix} x/w \\ y/w \\ z/w \end{bmatrix}$ Les coordonnées euclidiennes

Pas 1-à-1!

LA MATRICE AUGMENTÉE

La transformation linéaire

Translation

$$\begin{bmatrix} m_{11} & m_{12} & m_{13} & b_x \\ m_{21} & m_{22} & m_{23} & b_y \\ m_{31} & m_{32} & m_{33} & b_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

LES TRANSFORMATIONS

Que font ces transformations affines 2D?

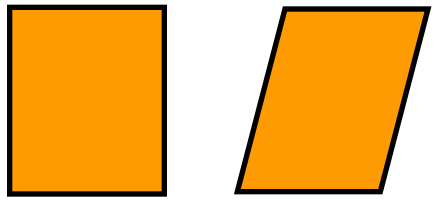
$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix}$$

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0.5 \end{bmatrix}$$

LES TRANSFORMATIONS

- Comment refléter un objet par une ligne connue (2D)?

- Quelle transformation fait ça?



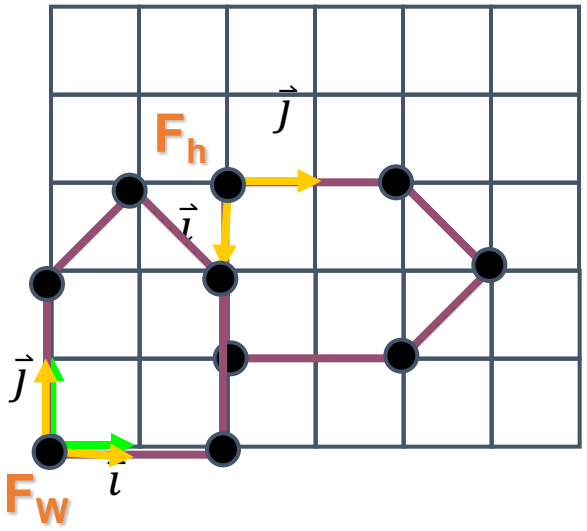
PENSER

- Que conserve chaque transformation?

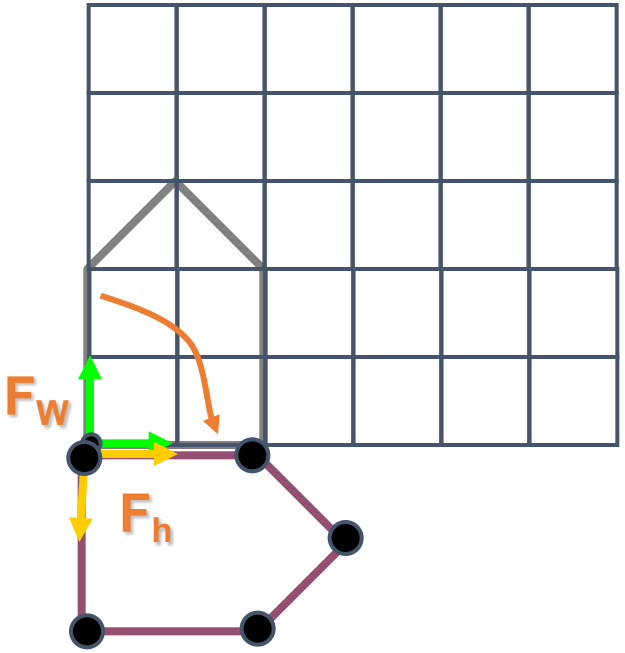
	lignes	lignes parallèles	distance	angles	normales	convexité
changement d'échelle						
rotation						
translation						
cisaillement						

LES TRANSFORMATIONS COMPOSÉES

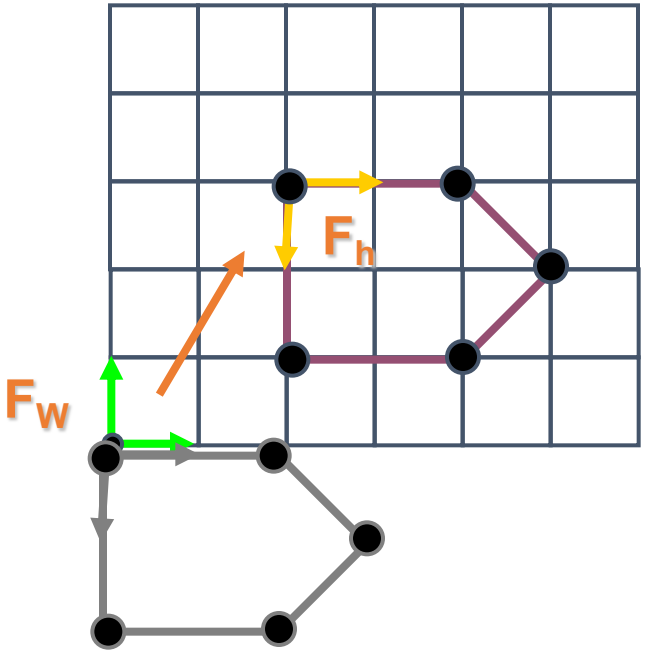
Si on veut



Rotate(z,-90)



Translate(2,3,0)



$$P_A = Rot(z, -90)P_h$$

$$P_W = Trans(2,3,0)P_A$$

$$P_W = Trans(2,3,0)Rot(z, -90)P_h$$

LES TRANSFORMATIONS COMPOSÉES

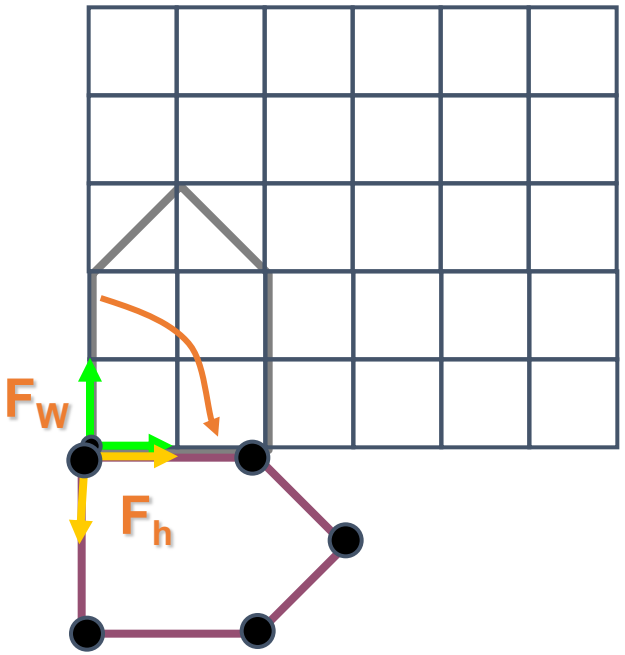
$$P_W = Trans(2,3,0)Rot(z,-90)P_h$$

- **Lire de droite à gauche** dans le système de coordonnées fixé
  - Déplacer l'objet
- **Lire de gauche à droite:** toutes les opérations sont dans le système de coordonnées local
  - Changer le système des coordonnées

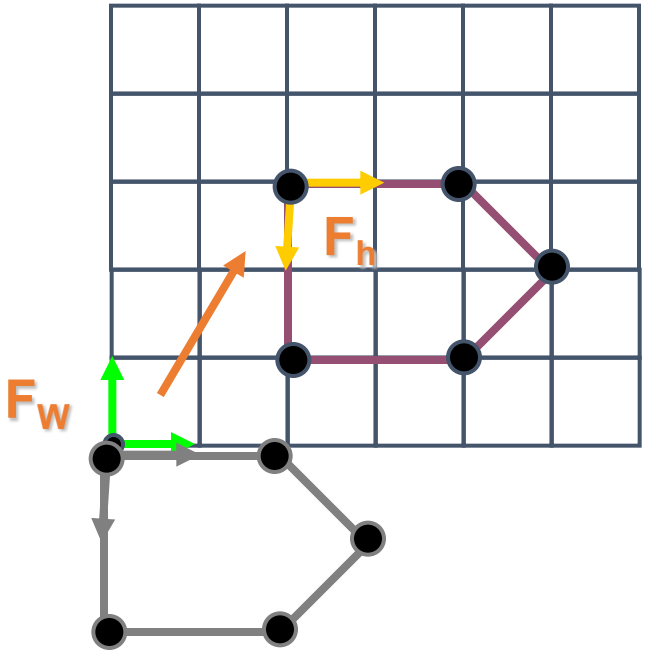
$$M_{MV} = Trans(2,3,0) \cdot M_{MV}$$
$$M_{MV} = Rot(z,-90)M_{MV}$$

LES TRANSFORMATIONS COMPOSÉES

Rotate(z,-90)



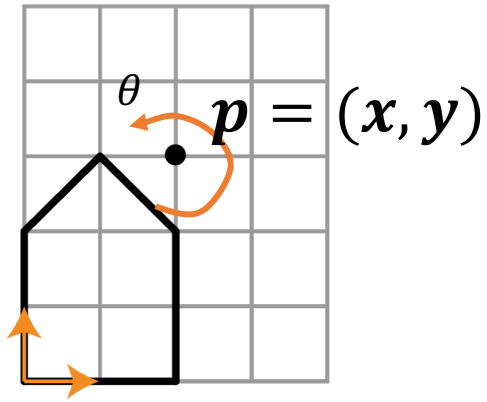
Translate(-3,2,0) dans le système de coordonnées local



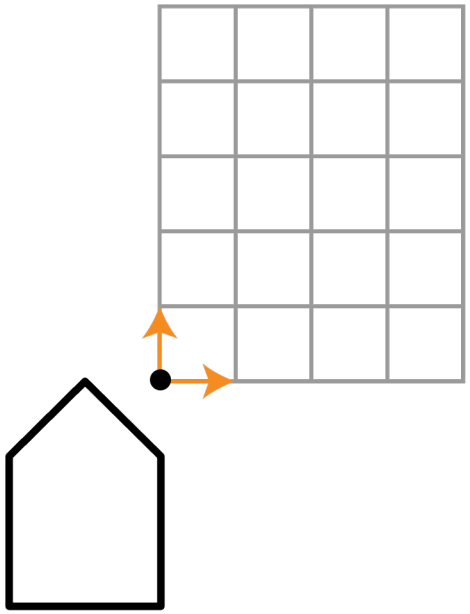
$$P_W = Rot(z, -90)Trans(-3,2,0)P_h$$

LA ROTATION AUTOUR D'UN POINT

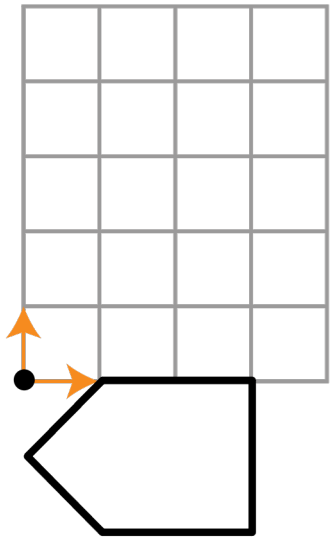
Tourner autour p de θ:



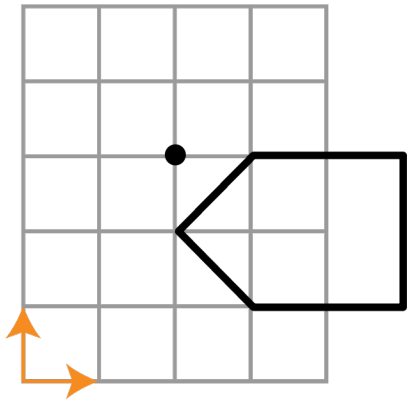
Déplacer p à l'origine



Tourner autour l'origine



Déplacer p en arrière

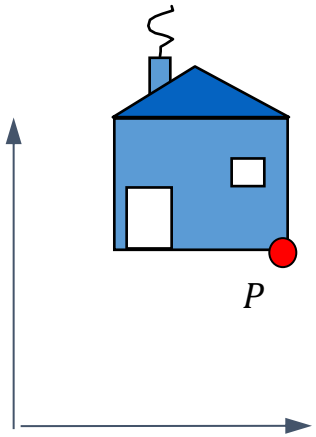


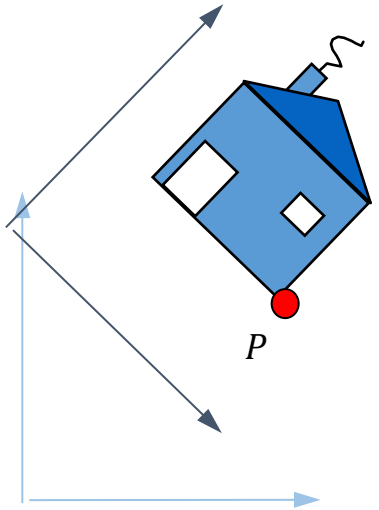
$$\mathbf{T}(x,y,z)\mathbf{R}(z,\theta)\mathbf{T}(-x,-y,-z)$$

TRANSFORMATION DE COORDONNÉES

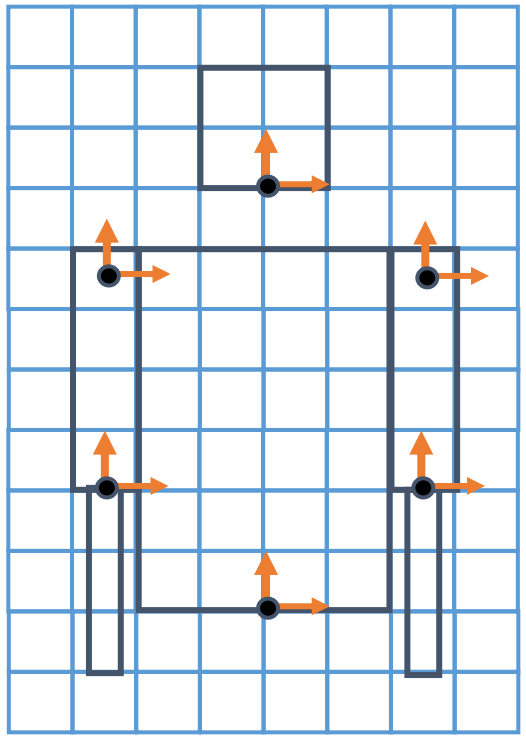
Les colonnes sont les nouveaux vecteurs de base (et la nouvelle origine)!

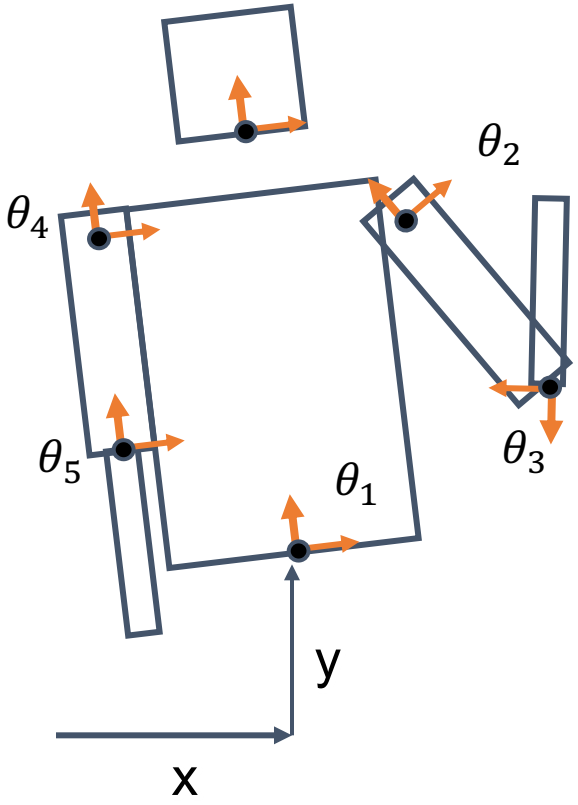
$$\begin{pmatrix} \cos\theta & -\sin\theta & p_x \cdot (1-\cos\theta) + p_y \cdot \sin\theta \\ \sin\theta & \cos\theta & p_y \cdot (1-\cos\theta) + p_x \cdot \sin\theta \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} v_x \\ v_y \\ 1 \end{pmatrix}$$





LA HIÉRARCHIE DE TRANSFORMATIONS





$$M_1 = Tr_{(x,y)} \cdot Rot\ \theta_1$$

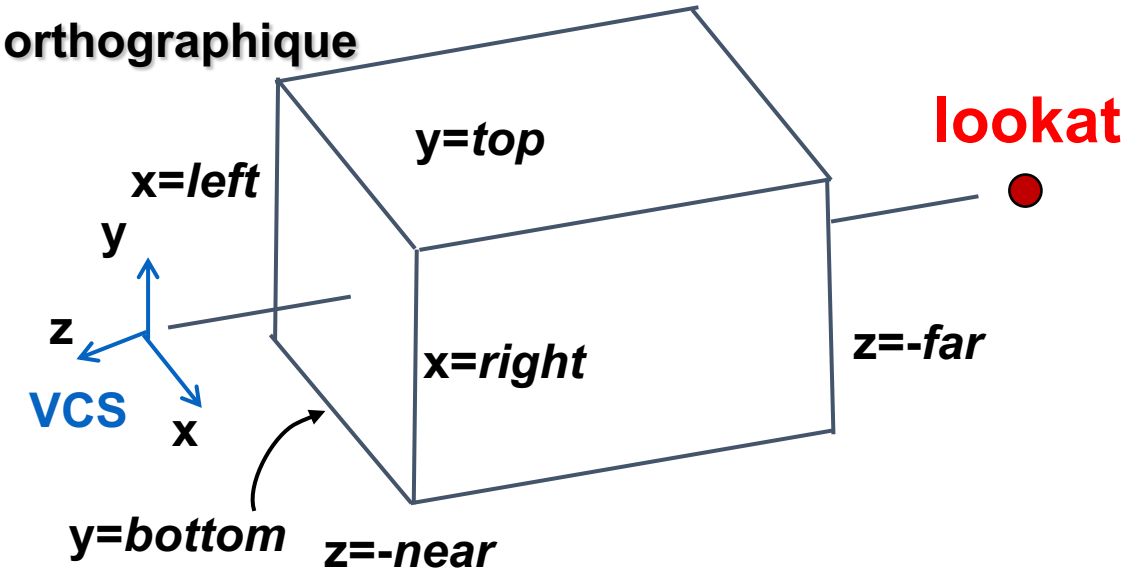
$$M_2 = M_1 \cdot Tr_{(2.5,5.5)} \cdot Rot\ \theta_2$$

$$M_3 = M_2 \cdot Tr_{(0,-3.5)} \cdot Rot\ \theta_3$$

LE VOLUME DE VUE PROJECTION ORTHOGRAPHIQUE

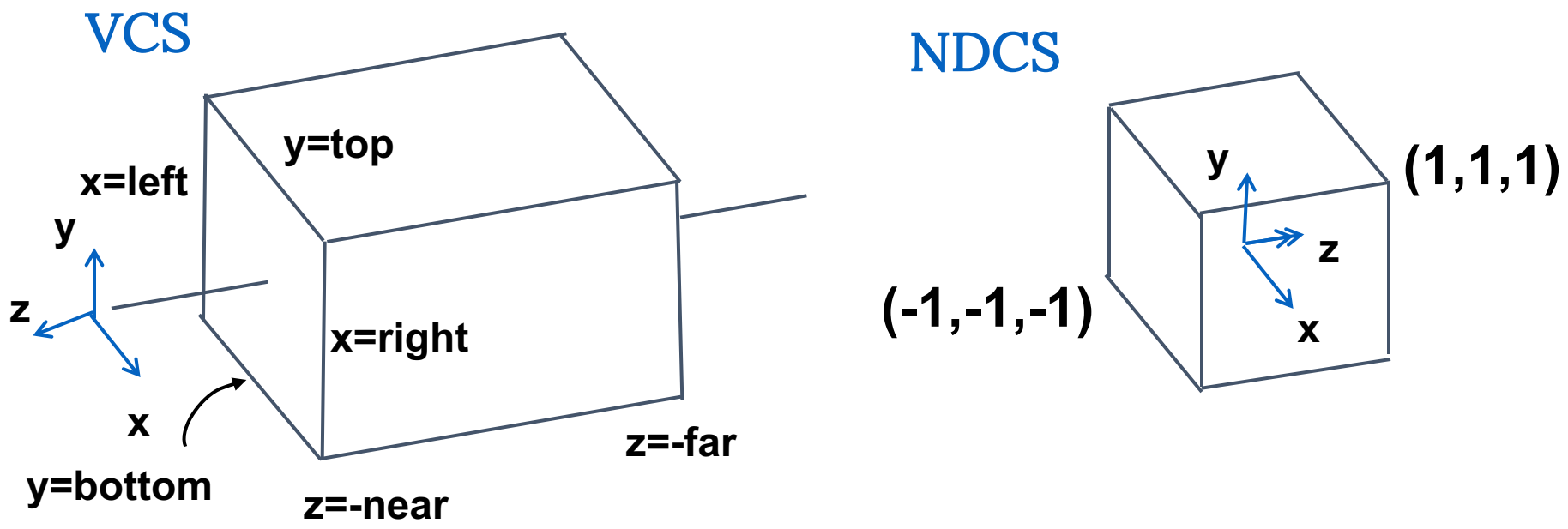
- Est une projection parallèle dans la direction perpendiculaire au plan de projection
- Il spécifie le champ de vue, utilisé pour le *clipping*
- Il limite le domaine de *z* stocké pour le test de profondeur

le volume de vue orthographique



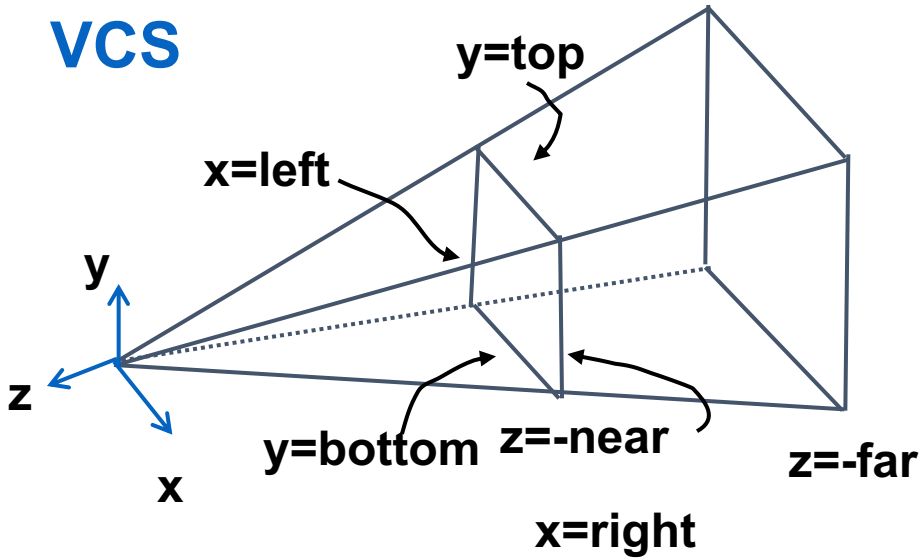
L'AXE *Z*

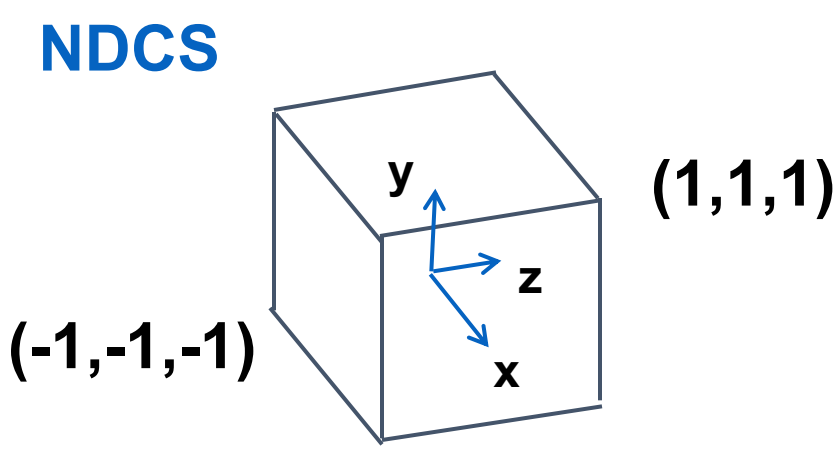
- Un flip de l'axe *z* change le *handedness* du système des coordonnées
  - RHS (règle de la main droite) avant la projection
    - les coordonnées de la vue/caméra, du monde
  - LHS (règle de la main gauche) après
    - clip, les coordonnées de l'appareil normalisées



LA MATRICE DE LA PROJECTION PERSPECTIVE OPENGL

Mapper (faire correspondre) la pyramide tronquée à un cube NDCS. Après, la coordonnée *z* sera ignorée. On doit changer l'échelle/déplacer/faire cisaillement → une transformation générique





LES LIGNES ET LES COURBES

- Les fonction explicites: les coordonnées sont des fonctions des autres

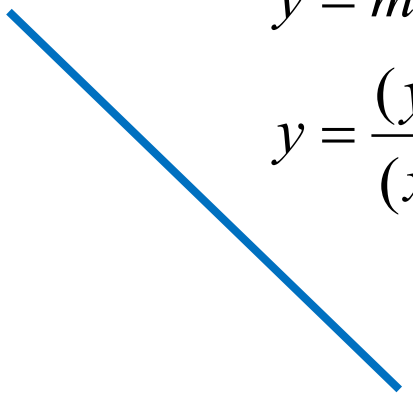
$$y = f(x)$$

$$z = f(x, y)$$

Line

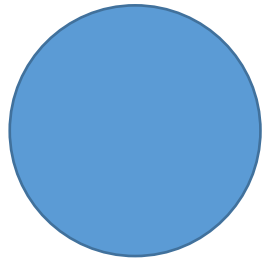
$$y = mx + b$$

$$y = \frac{(y_2 - y_1)}{(x_2 - x_1)}(x - x_1) + y_1$$



Circle

$$y = \pm\sqrt{r^2 - x^2}$$



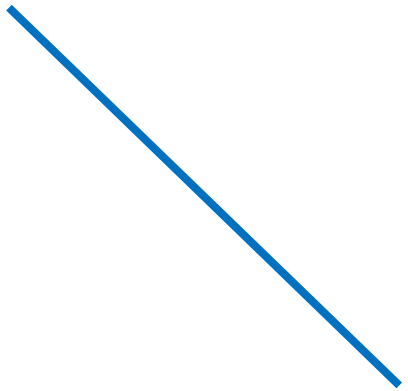
LES LIGNES ET LES COURBES

Les fonctions paramétriques: toutes les coordonnées sont définies en fonction des valeurs de paramètres

$$(x, y) = (f_1(t), f_2(t))$$

$$(x, y, z) = (f_1(u, v), f_2(u, v), f_3(u, v))$$

Une ligne



$$x(t) = x_1 + t(x_2 - x_1)$$

$$y(t) = y_1 + t(y_2 - y_1)$$

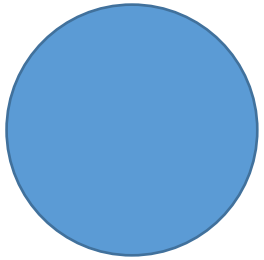
$$t \in [0,1]$$

Un Cercle

$$x(\theta) = r\cos(\theta)$$

$$y(\theta) = r\sin(\theta)$$

$$\theta \in [0,2\pi]$$



LES LIGNES ET LES COURBES

• Les fonctions implicites: l'objet est défini par les racines de la fonction

$$\{(x,y) : F(x,y) = 0\}$$

$$\{(x,y,z) : F(x,y,z) = 0\}$$

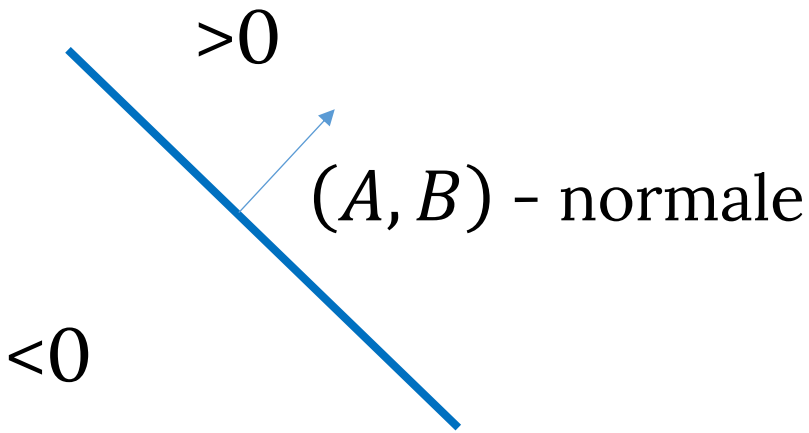
- Diviser l'espace

$$\{(x,y) : F(x,y) > 0\}, \{(x,y) : F(x,y) = 0\}, \{(x,y) : F(x,y) < 0\}$$

LES FONCTIONS IMPLICITES

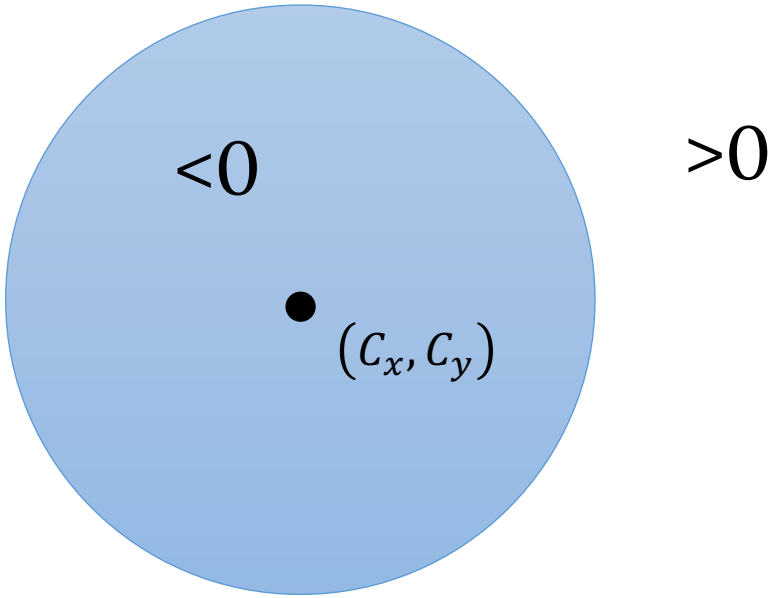
Une ligne

$$Ax + By + C = 0$$



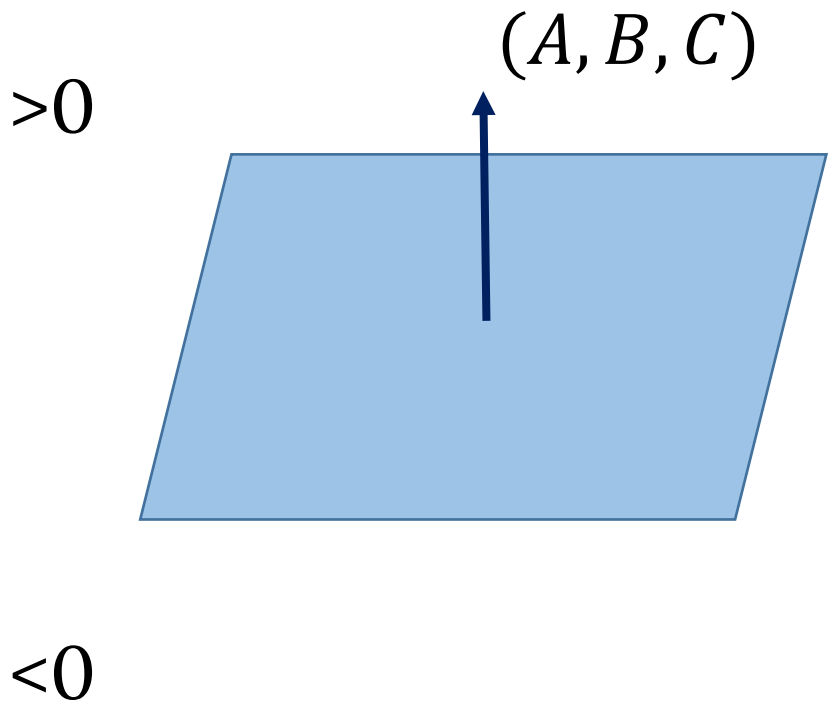
Un cercle

$$(x - C_x)^2 + \left(y - C_y\right)^2 - r^2 = 0$$



LE PLAN - IMPLICITE

$$Ax + By + Cz + D = 0$$

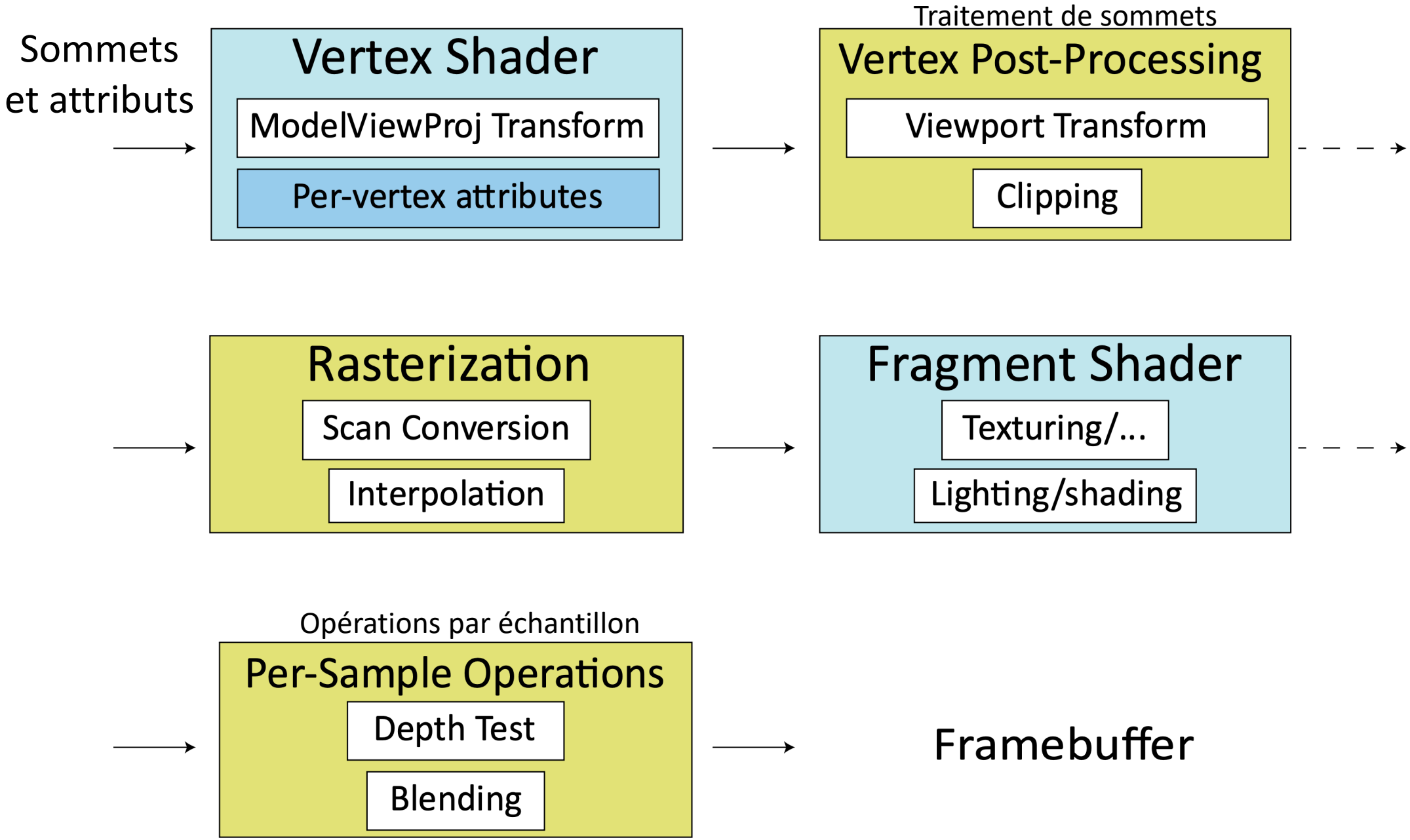


UNE FONCTION IMPLICITE ARBITRAIRE

$$F(x, y, z) = 0$$

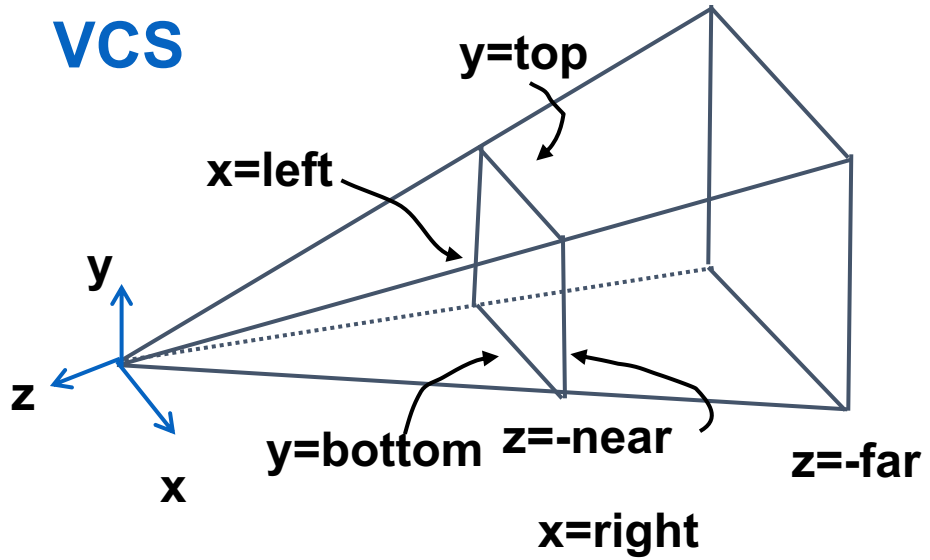
$$n(x, y, z) = \nabla F(x, y, z) = \begin{pmatrix} \partial F(x, y, z)/\partial x \\ \partial F(x, y, z)/\partial y \\ \partial F(x, y, z)/\partial z \end{pmatrix}$$

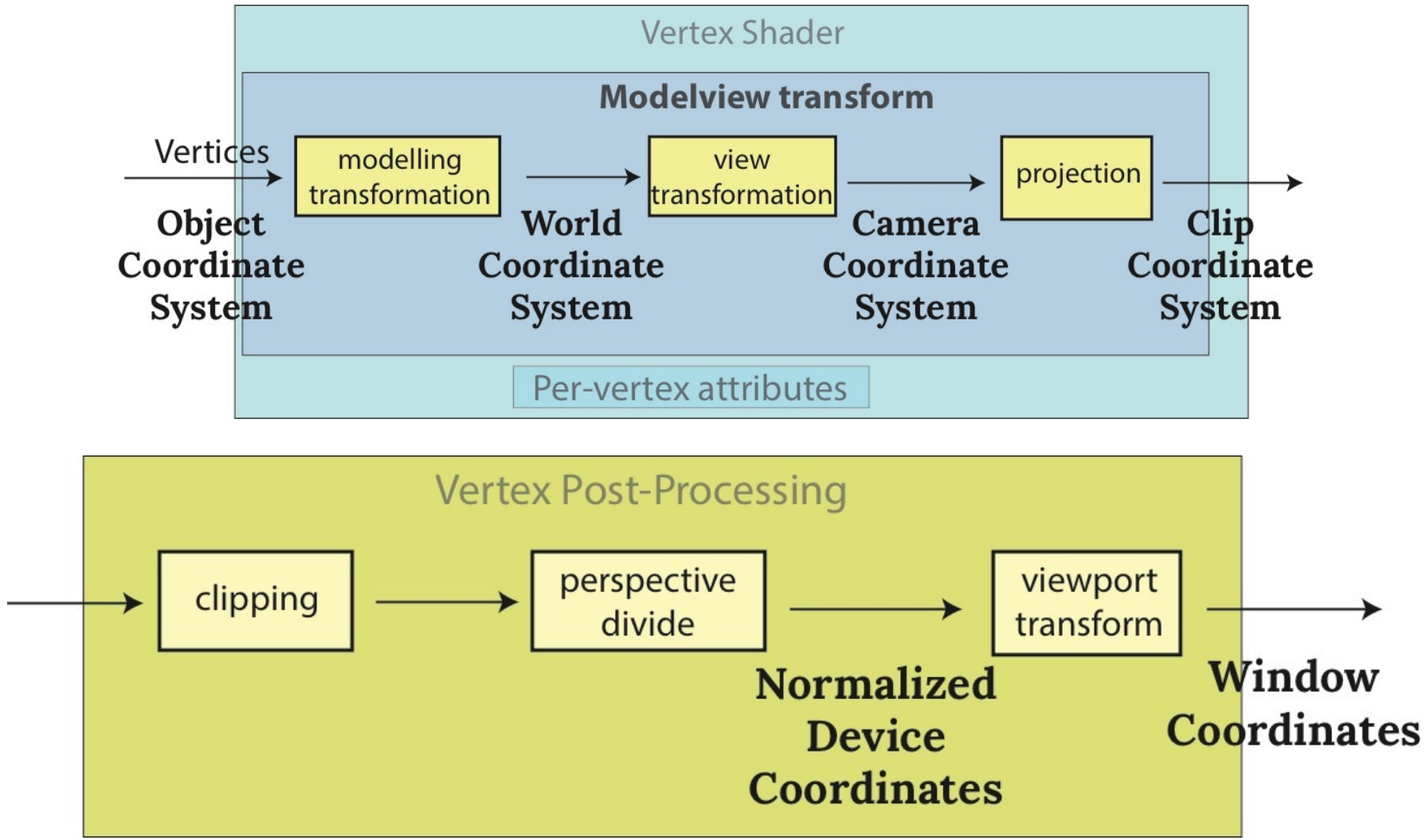
PIPELINE: PLUS DE DÉTAILS



CLIPPING

- Il faut découper tout ce qui est hors du volume de vue
- Hors du plan gauche/droit, supérieur/inférieur
- Et plus important, avant/arrière:




Vertex Shader
Modelview transform
Vertices
modelling transformation
view transformation
projection
Object Coordinate System
World Coordinate System
Camera Coordinate System
Clip Coordinate System
Per-vertex attributes
Vertex Post-Processing
clipping
perspective divide
viewport transform
Normalized Device Coordinates
Window Coordinates

LA MATRICE DE LA FENÊTRE (*VIEWPORT)*

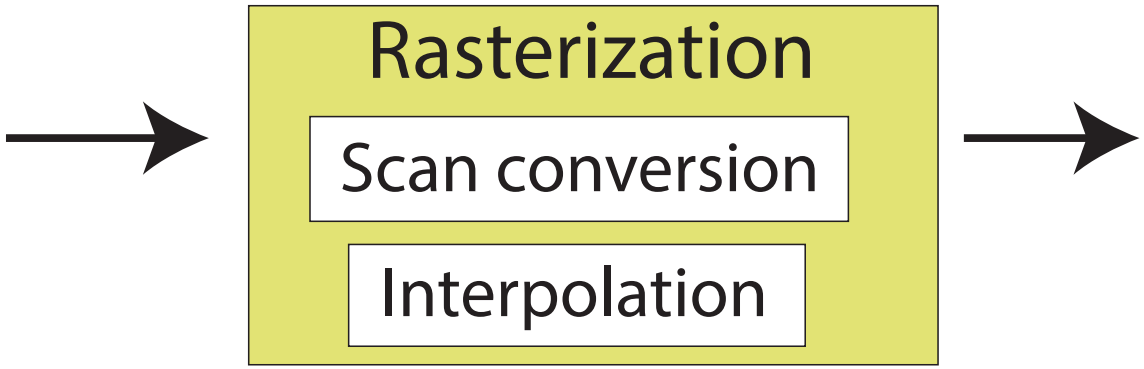
- Qu'est-ce qu'elle fait?

$$\begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{W}{2} & 0 & 0 & \frac{W-1}{2} \\ 0 & \frac{H}{2} & 0 & \frac{H-1}{2} \\ 0 & 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_n \\ y_n \\ z_n \\ 1 \end{bmatrix}$$

RASTERIZATION

- C'est une partie du pipeline fixe

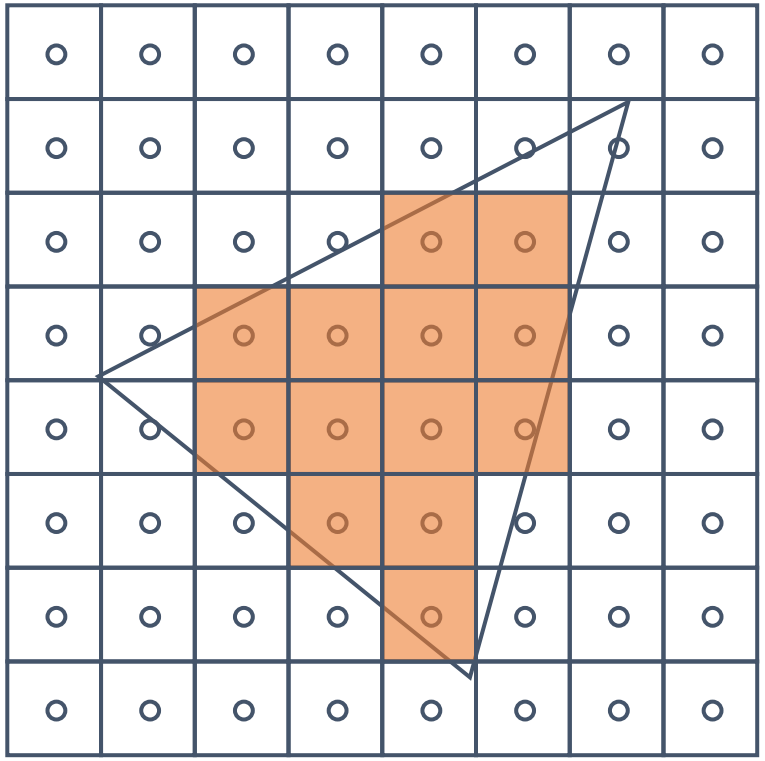
- Les données en entrée: tous les polygone clippés
- Les données en sortie: les fragments (avec les **varying** variables interpolées)

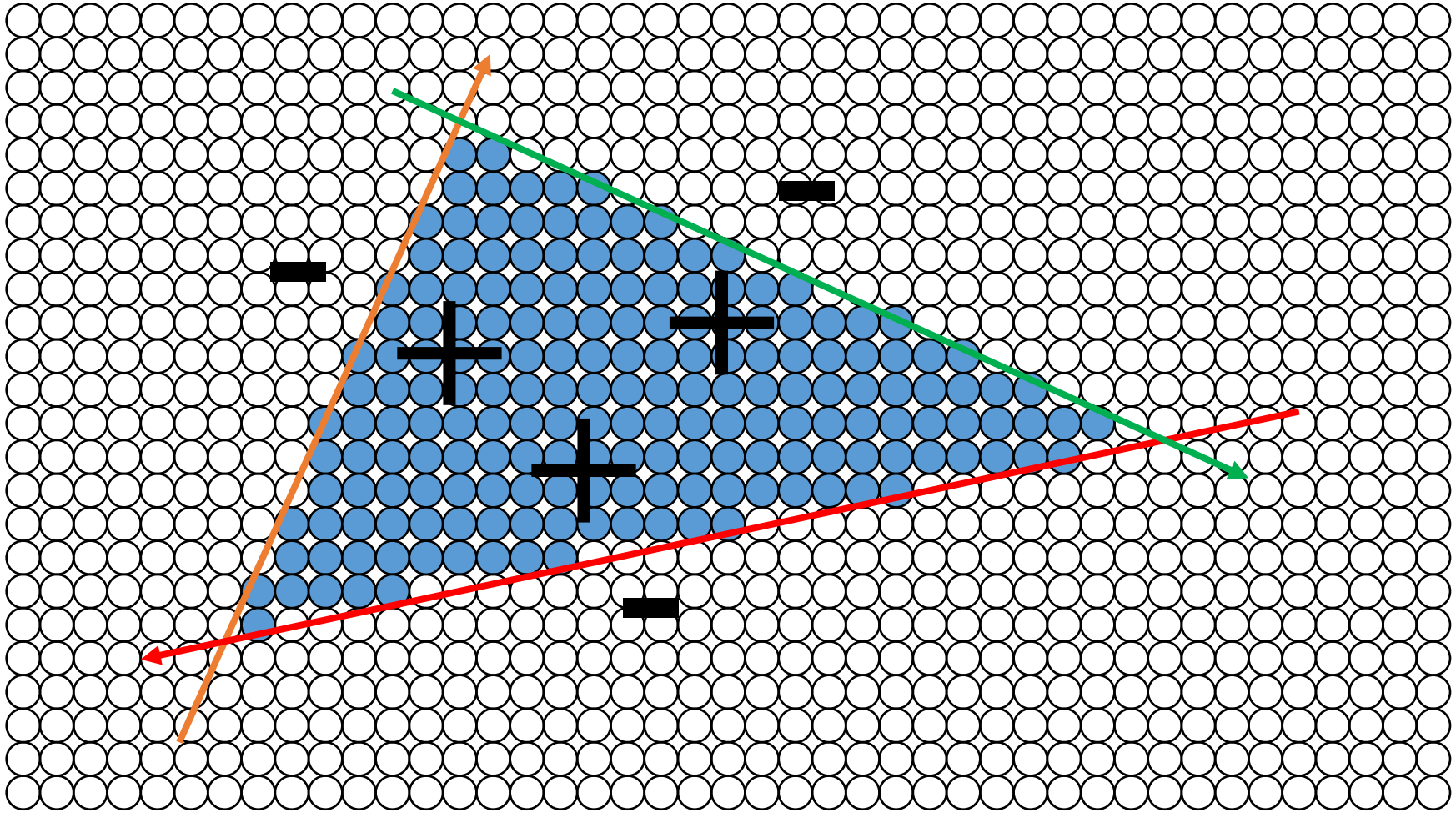


COMMENT VÉRIFIER SI UN PIXEL EST À L'INTÉRIEUR?

Un point est à l'intérieur ⇔

$$A_i x + B_i y + C > 0, i = 1, \dots, 3$$





SCANLINE: CODE

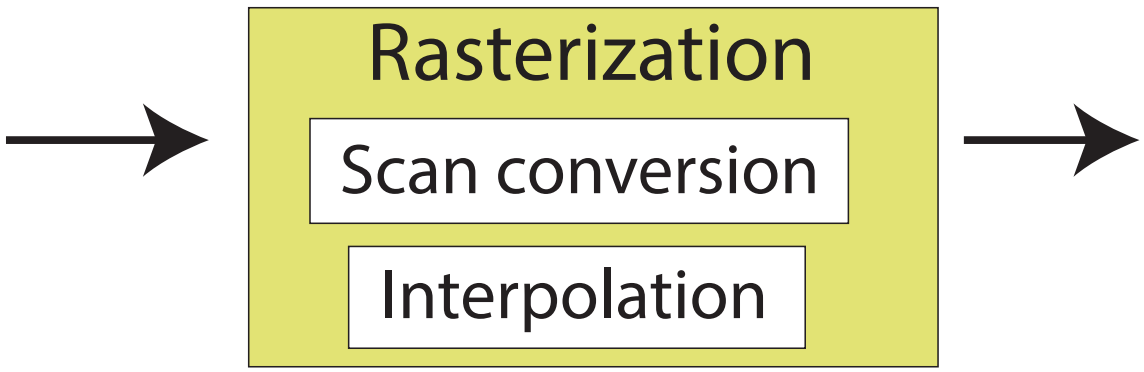
```
findBoundingBox(xmin, xmax, ymin, ymax);
setupEdges (a0,b0,c0,a1,b1,c1,a2,b2,c2);

for (int y = yMin; y <= yMax; y++) {
  for (int x = xMin; x <= xMax; x++) {
    float e0 = a0*x + b0*y + c0;
    float e1 = a1*x + b1*y + c1;
    float e2 = a2*x + b2*y + c2;
    if (e0 > 0 && e1 > 0 && e2 > 0)
        Image[x][y] = TriangleColor;
  }
}
```

SCAN CONVERSION

- What are problems of scan conversion?
- How to find a bounding box?
- How to scan-convert an arbitrary polygon?

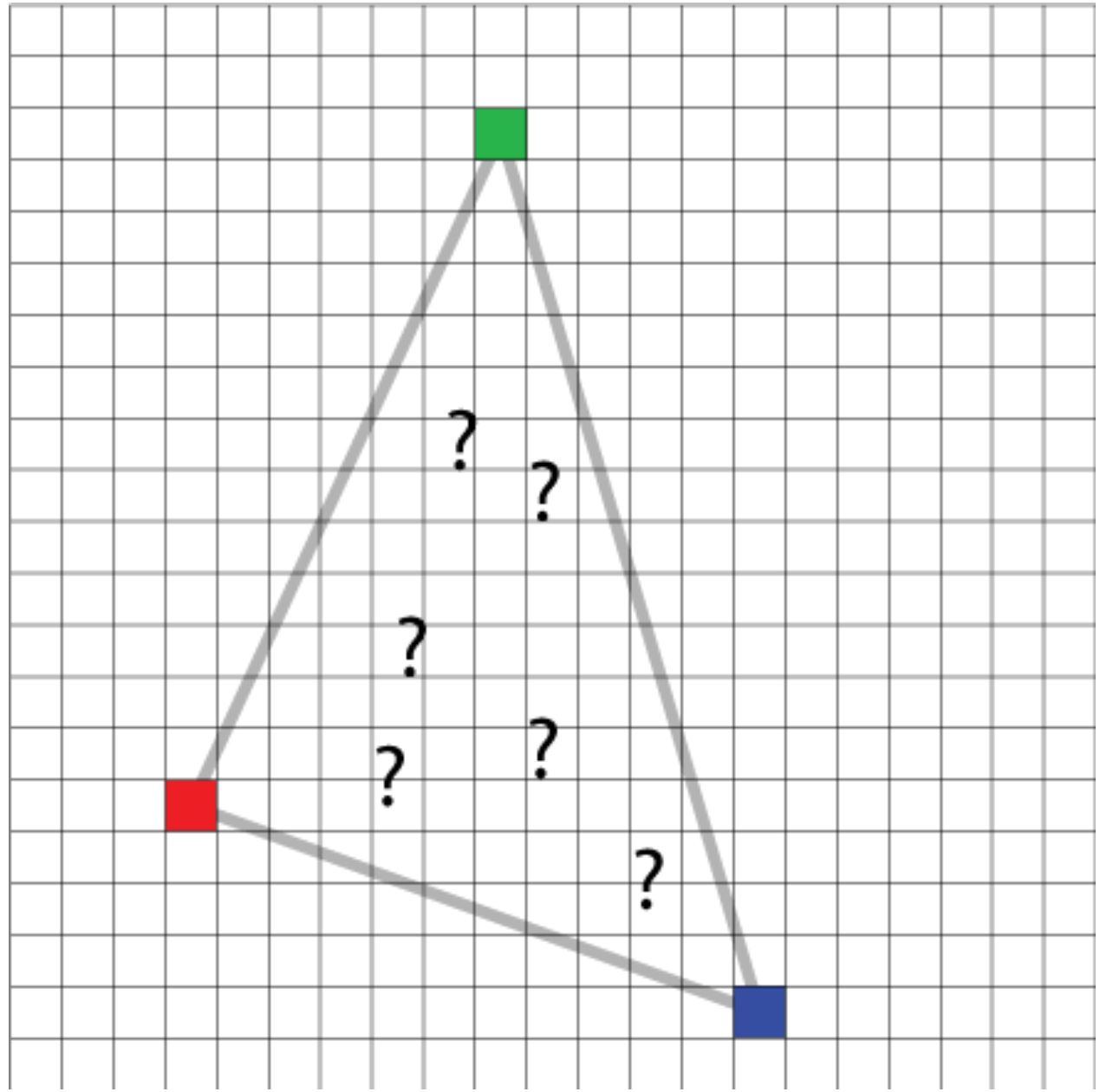
INTERPOLATION

- What does it do?



L'INTERPOLATION : CALCULER LES VALEURS ENTRE CELLES QU'ON CONNAÎT

- On interpole:
  - z
  - r, g, b – les composants de la couleur
  - u, v - les coordonnées de la texture
  - (n_x, n_y, n_x) – les composants de la normale
- Les méthodes suivantes sont équivalentes:
  - Les coordonnées barycentriques
  - L'interpolation bilinéaire
  - L'interpolation du plan

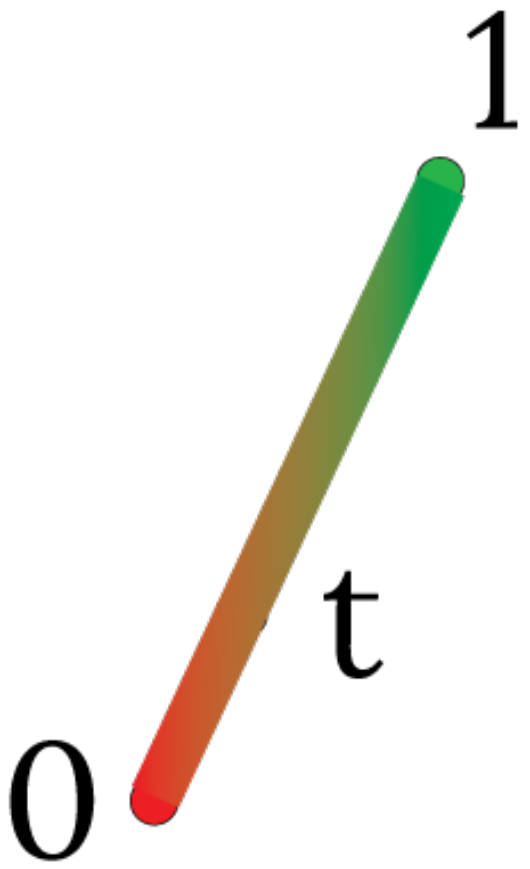


PLUS FACILE:

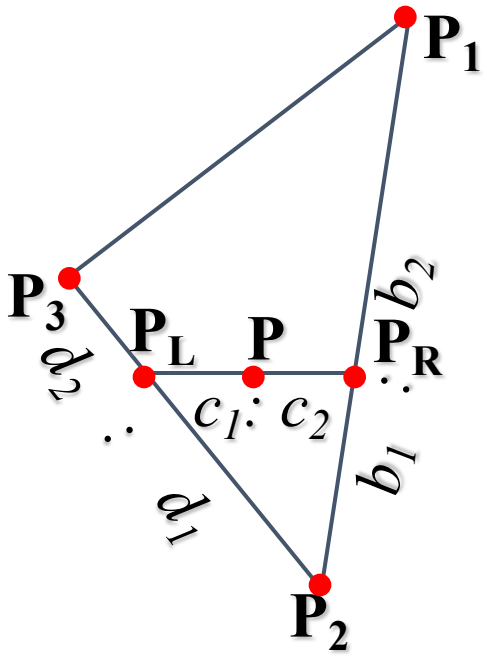
Comment interpoler la couleur entre les deux sommets?

$$c(t) = c(0) \cdot (1 - t) + c(1) \cdot t$$

Interpolation linéaire



L'INTERPOLATION BILINÉAIRE



$$P = \frac{c_2}{c_1 + c_2} \cdot P_L + \frac{c_1}{c_1 + c_2} \cdot P_R$$

$$P_L = \frac{d_2}{d_1 + d_2} P_2 + \frac{d_1}{d_1 + d_2} P_3$$

$$P_R = \frac{b_2}{b_1 + b_2} P_2 + \frac{b_1}{b_1 + b_2} P_1$$

$$P = \frac{c_2}{c_1 + c_2}\left(\frac{d_2}{d_1 + d_2} P_2 + \frac{d_1}{d_1 + d_2} P_3\right) + \frac{c_1}{c_1 + c_2}\left(\frac{b_2}{b_1 + b_2} P_2 + \frac{b_1}{b_1 + b_2} P_1\right)$$

LES COORDONNÉES BARYCENTRIQUES

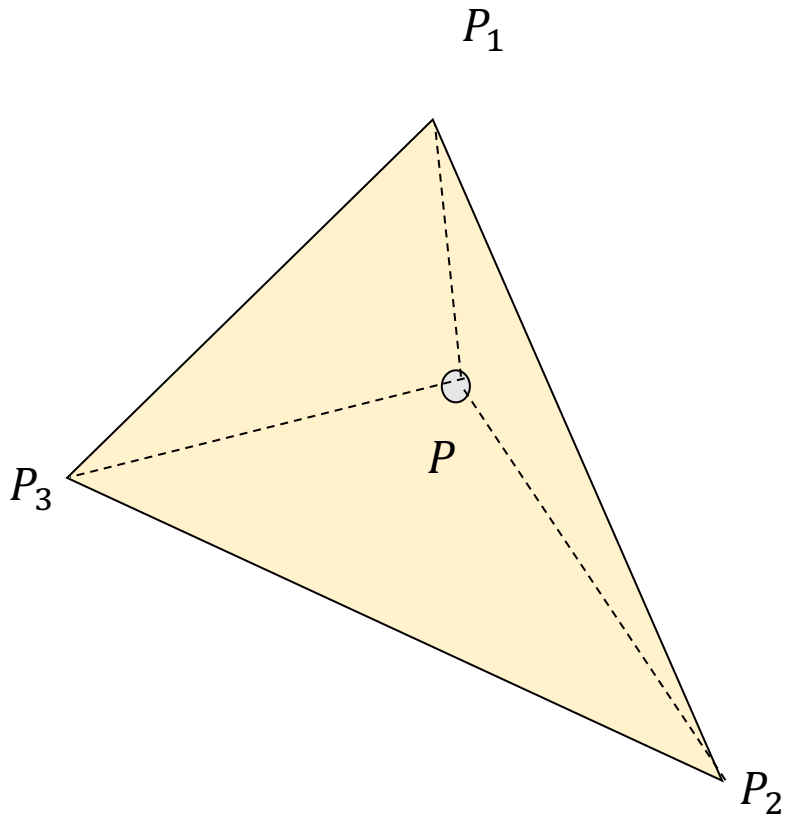
- L'aire

$$A = \frac{1}{2}\left\|\overrightarrow{P_1P}_2 \times \overrightarrow{P_1P_3}\right\|$$

- Les coordonnées barycentriques:

$$a_1 = A_{P_2P_3P}/A, a_2 = A_{P_3P_1P}/A,$$
$$a_3 = A_{P_1P_2P}/A,$$
$$P = a_1P_1 + a_2P_2 + a_3P_3$$

$$f(P) = a_1f(P_1) + a_2f(P_2) + a_3f(P_3)$$

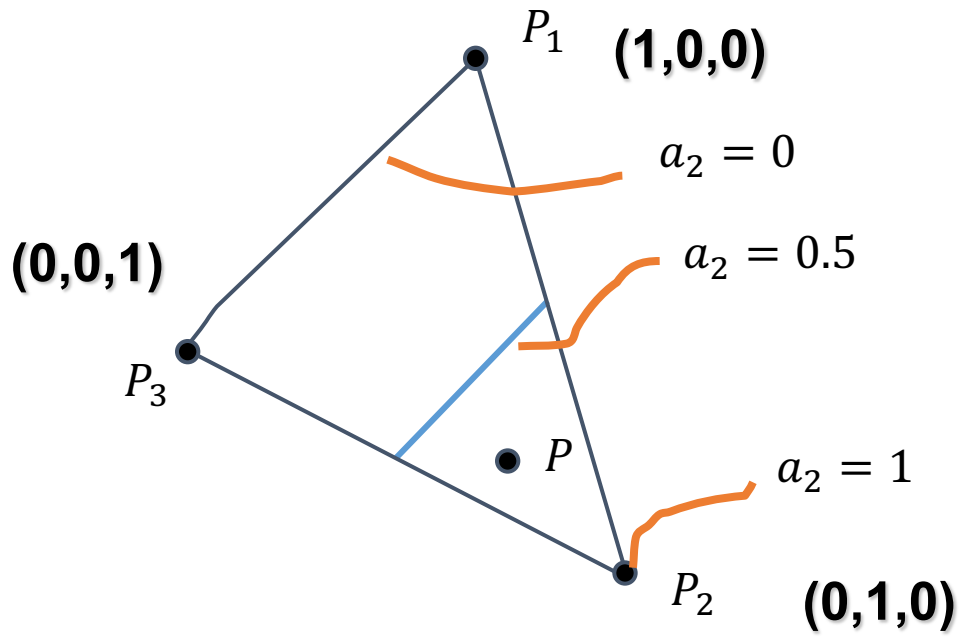


CHAQUE POINT À L'INTÉRIEUR DU TRIANGLE EST

- Une combinaison pondérée (affine) des sommets

$$P = a_1 \cdot P_1 + a_2 \cdot P_2 + a_3 \cdot P_3$$

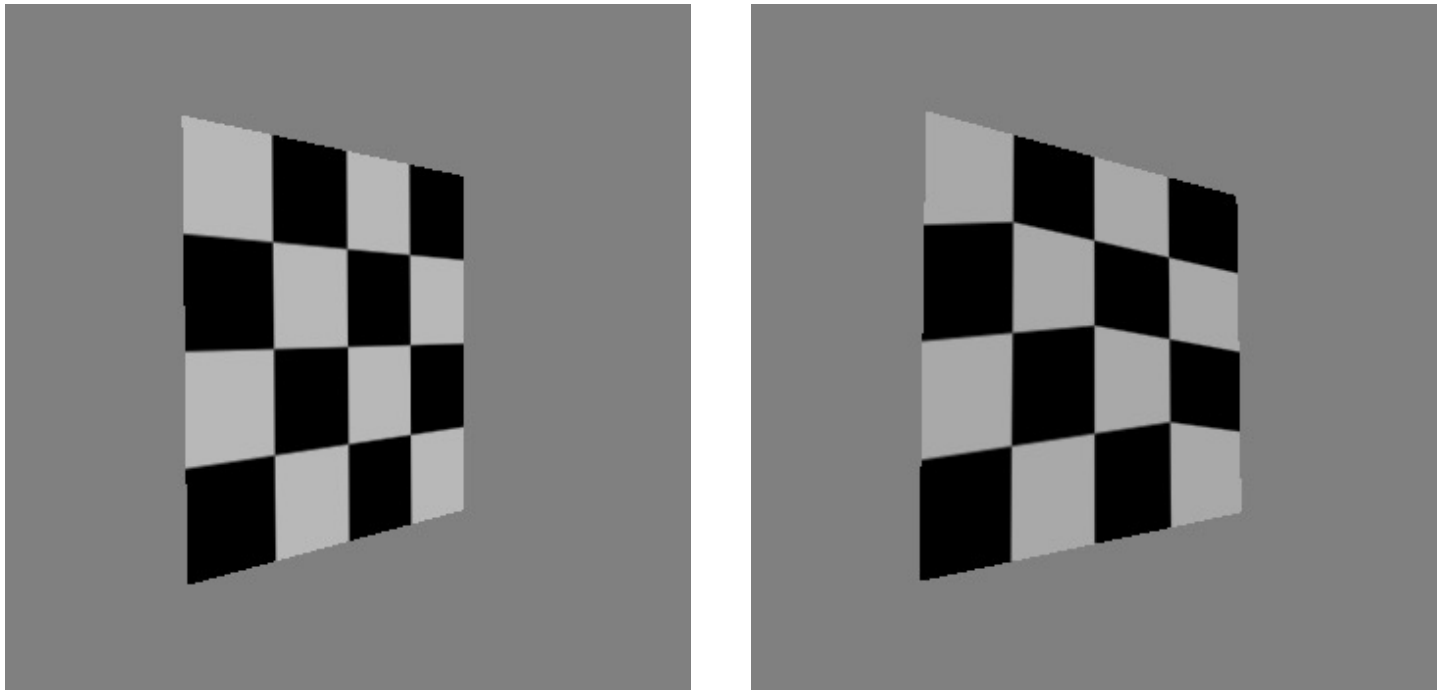
$$a_1 + a_2 + a_3 = 1$$
$$0 \leq a_1, a_2, a_3 \leq 1$$



TEXTURE MAPPING

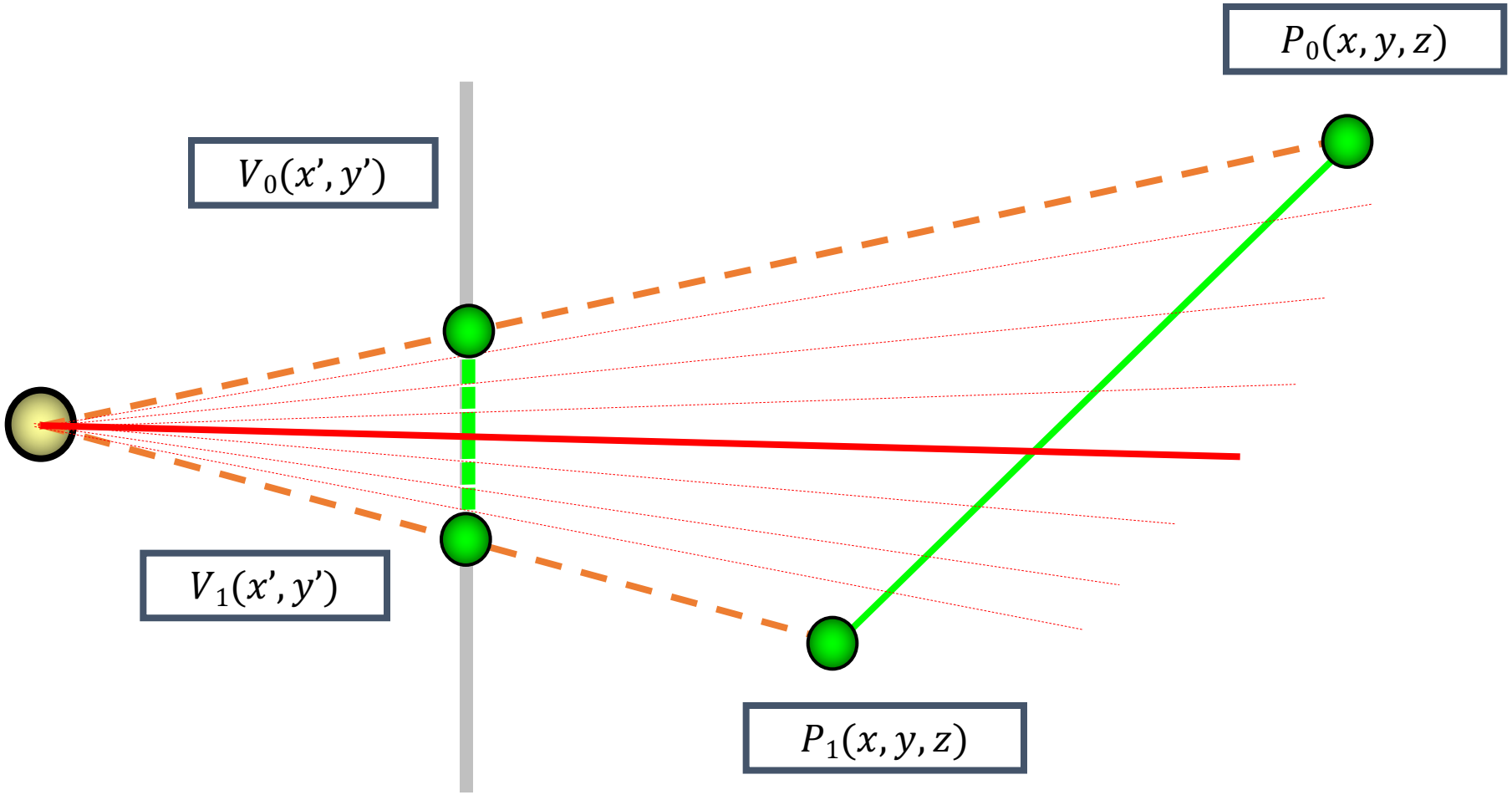
Après la projection perspective, l'interpolation linéaire dans l'espace de l'affichage est incorrecte

- Pour les textures, aussi bien que pour les couleurs, shading, etc.



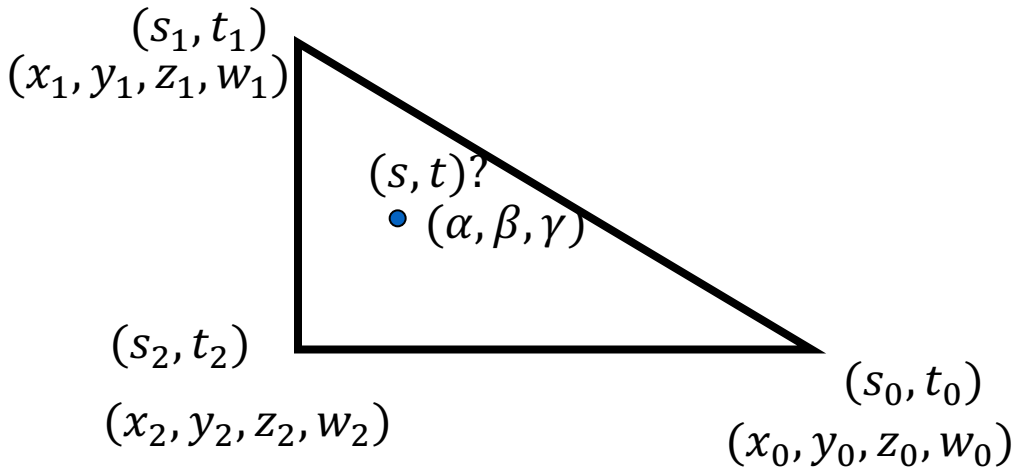
INTERPOLATION: L'ESPACE DE L'AFFICHAGE VS DU MONDE

- Normalement, on ignore le problème pour le *shading*, mais pour les textures les artefacts sont évidents



L'INTERPOLATION APRÈS LA PERSPECTIVE

- α, β, γ : les coordonnées barycentriques (2D) du point P
- s_0, s_1, s_2 : les coordonnées dans l'espace de la texture
- w_0, w_1, w_2 : les coordonnées homogènes des vecteurs



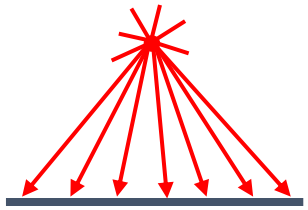
$$s = \frac{\alpha \cdot s_0/w_0 + \beta \cdot s_1/w_1 + \gamma \cdot s_2/w_2}{\alpha/w_0 + \beta/w_1 + \gamma/w_2}$$

- La même chose pour t

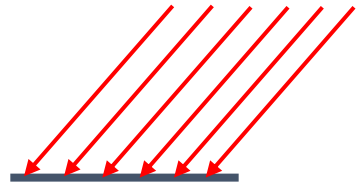
La dérivation (les triangles similaires):
https://www.comp.nus.edu.sg/~lowkl/publications/lowk_persp_interp_techrep.pdf

LES TYPES DE SOURCES DE LUMIIÈRE

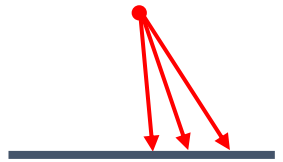
- Lumière ponctuelle
  - La lumière provient d'un point
  - définie par la position seulement



- Lumière directionnelle (=la lumière ponctuelle à l'infinité)
  - Les rayons sont parallèles
  - Les rayons frappent la surface au même angle
  - définie par la direction seulement



- Lumière ponctuelle de type *spotlight*
  - Une lumière ponctuelle mais avec les angles limités
  - définie par la position, la direction et l'intervalle des angles



LUMIÈRES

- La lumière a une couleur
- Elle interagit avec la couleur de la surface (r, g, b)

$$I = I_a k_a$$

$$I_a = (I_{ar}, I_{ag}, I_{ab})$$

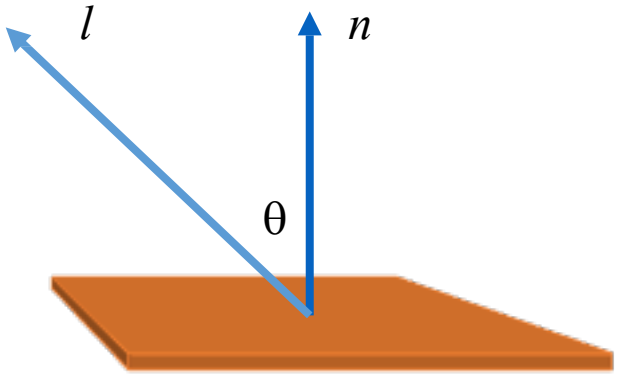
$$k_a = (k_{ar}, k_{ag}, k_{ab})$$

$$I = (I_r, I_g, I_b) = (I_{ar}k_{ar}, I_{ag}k_{ag}, I_{ab}k_{ab})$$

- La lumière bleue sur la surface blanche?
- La lumière bleue sur la surface rouge?

CALCULER LA RÉFLEXION DIFFUSE

Elle dépende de l'angle d'incidence: l'angle entre la normale est la direction de la lumière



$$I_{diffuse} = k_d I_{light} cos\theta = k_d I_{light}(n \cdot l)$$

Les variables suivantes sont de scalaires (pour les niveaux de gris) ou des triplés (la couleur)

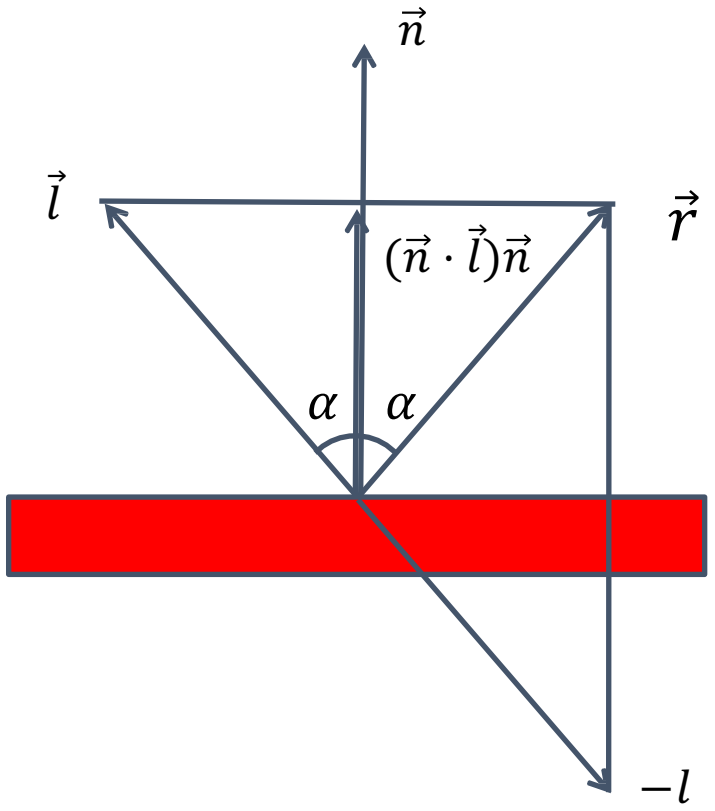
- k_d: un coefficient diffus, la couleur de surface
- I_{light}: l'intensité lumineuse entrante
- $I_{diffuse}$: l'intensité lumineuse sortante

NB: Il faut toujours normaliser les vecteurs en *shading*

- $\boldsymbol{n}, \boldsymbol{l}$ doivent avoir une longueur 1

LA PHYSIQUE DE LA RÉFLEXION SPÉCULAIRE

- La géométrie de la réflexion parfaite (spéculaire)
  - La loi de Snell-Descartes

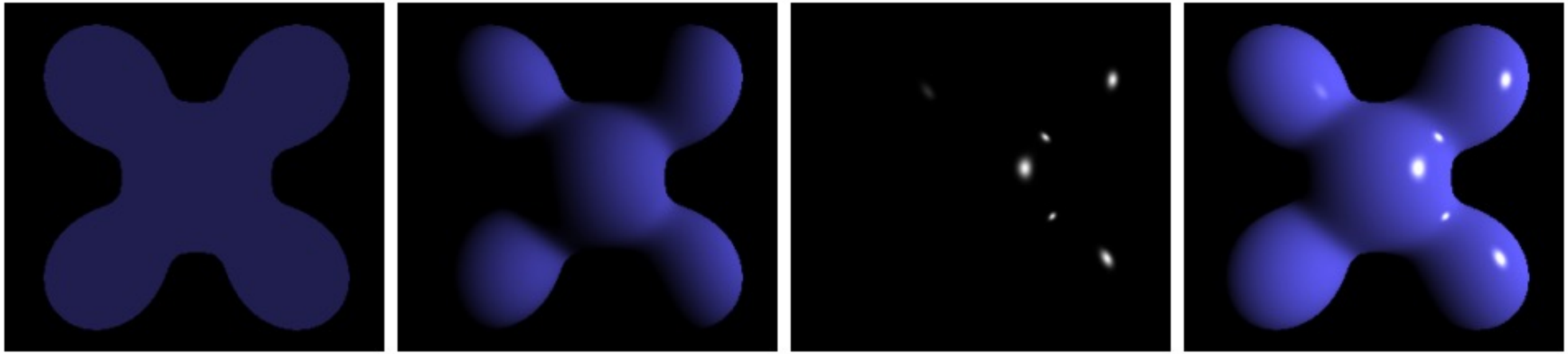


$$\vec{r} = -\vec{l} + 2(\vec{n} \cdot \vec{l})\vec{n}$$

L'ÉQUATION D'ÉCLAIRAGE (PHONG)

- Si on ajoute la composante de la lumière ambiante:

$$I_a k_a + \sum_p I_p (k_d (n \cdot l_p) + k_s (r_p \cdot v)^n)$$



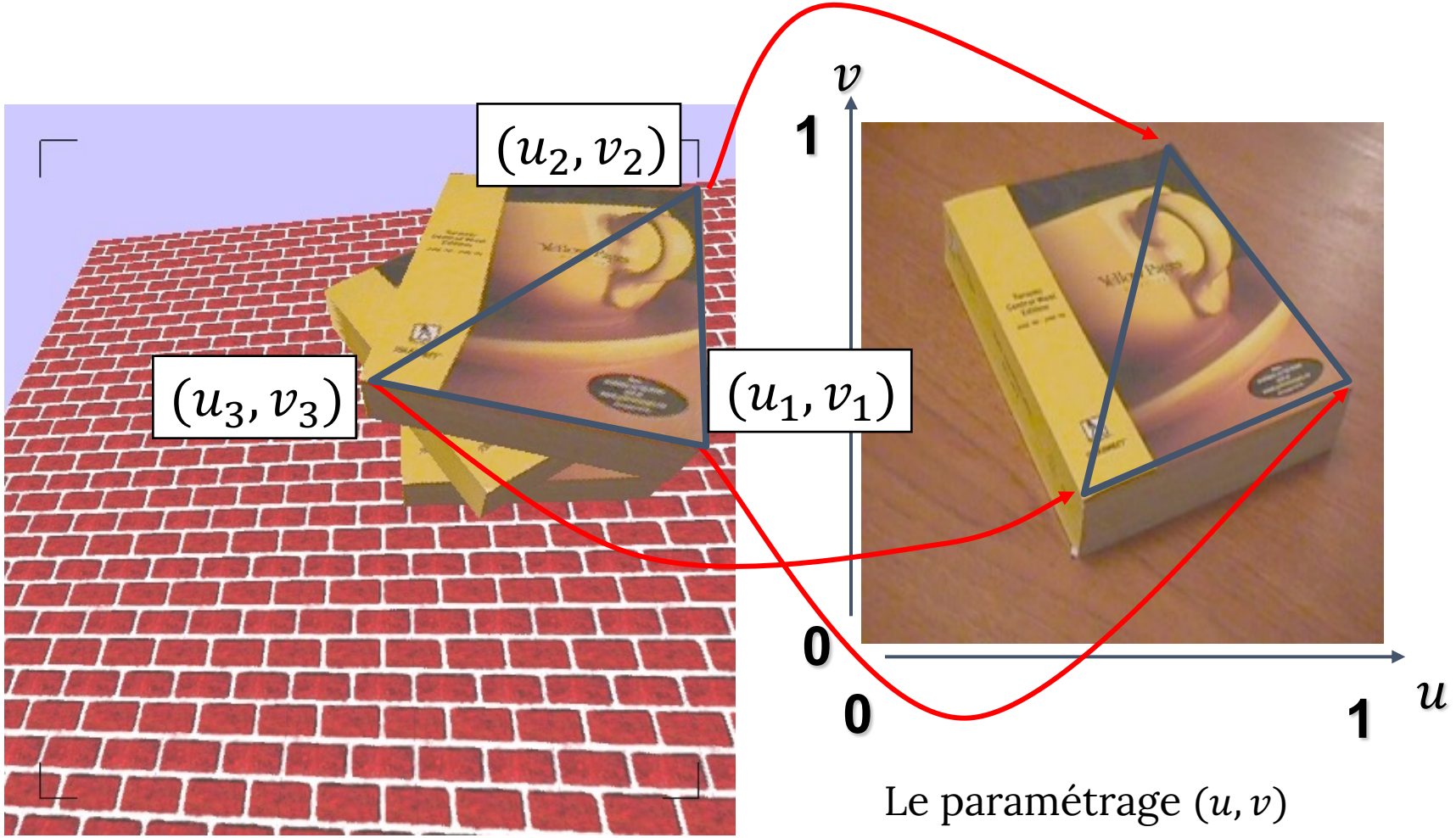
QUAND UTILISER UN MODÈLE D'ÉCLAIRAGE?

Par polygone
"l'ombrage plat"

Par sommet
"l'ombrage
de Gouraud"

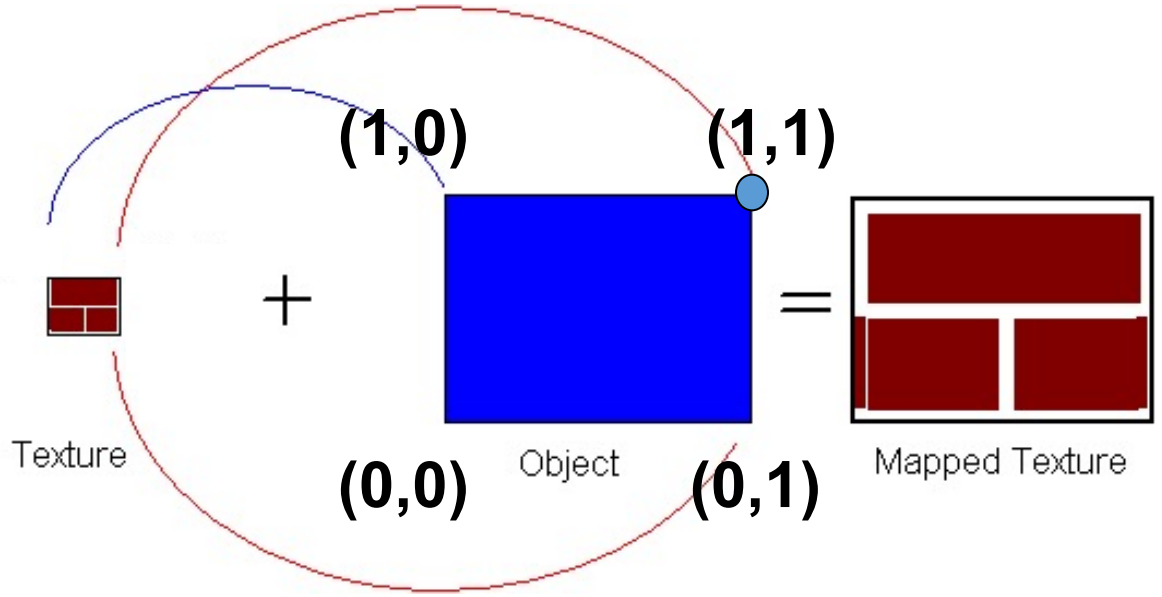
Par pixel/fragment
"per pixel lighting"
"l'ombrage de
Phong"

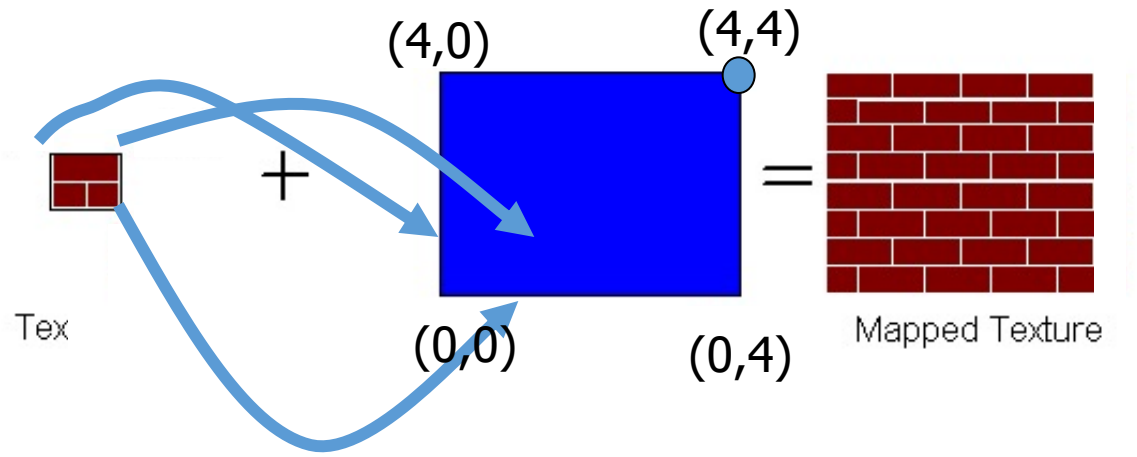
TEXTURE MAPPING



Le paramétrage (u, v)

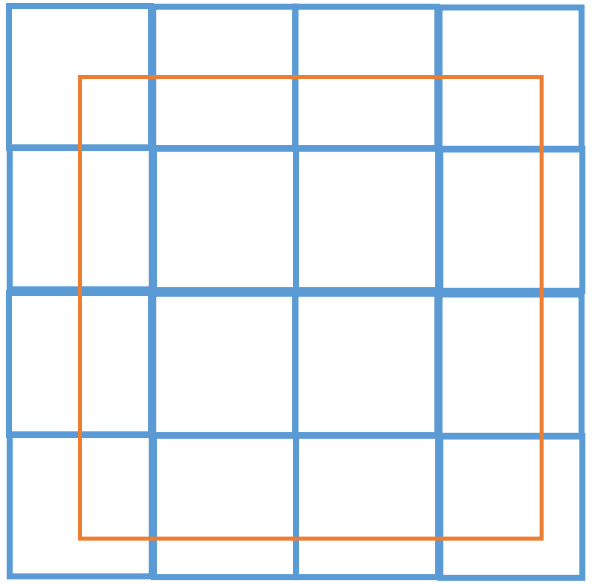
TILED TEXTURE MAP



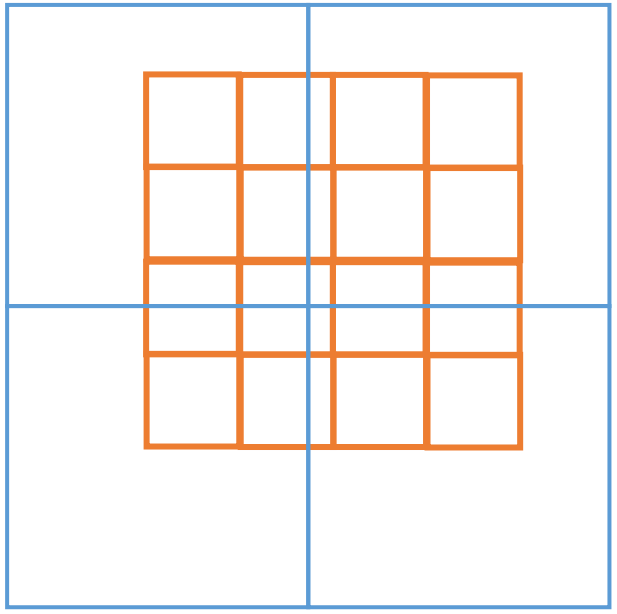


LA RECONSTRUCTION

- Comment faire avec:
  - Les pixels qui sont beaucoup plus grands que les texels?
    - La minification

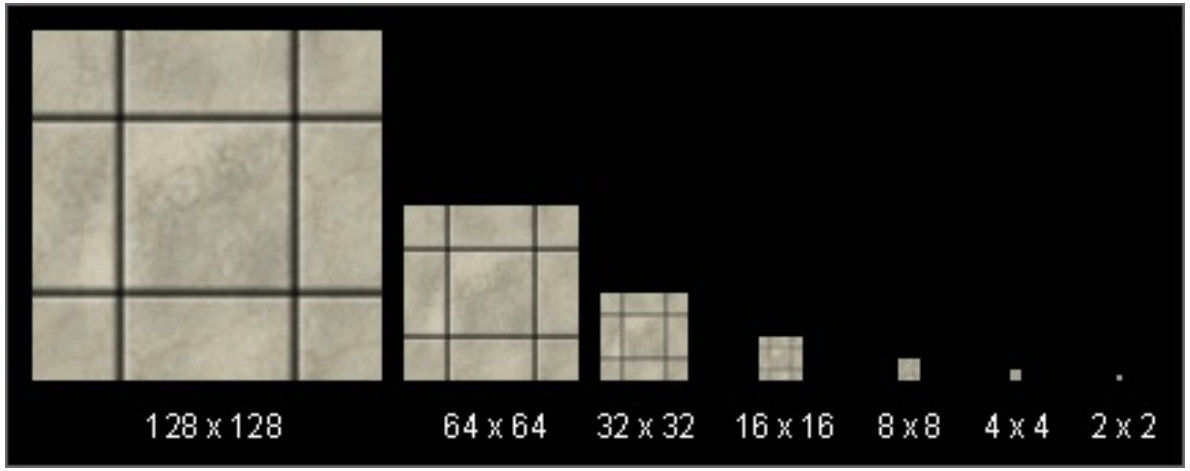


  - Les pixels qui sont beaucoup plus petits que les texels ?
    - La magnification

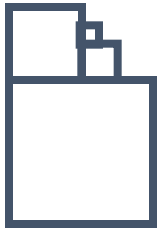


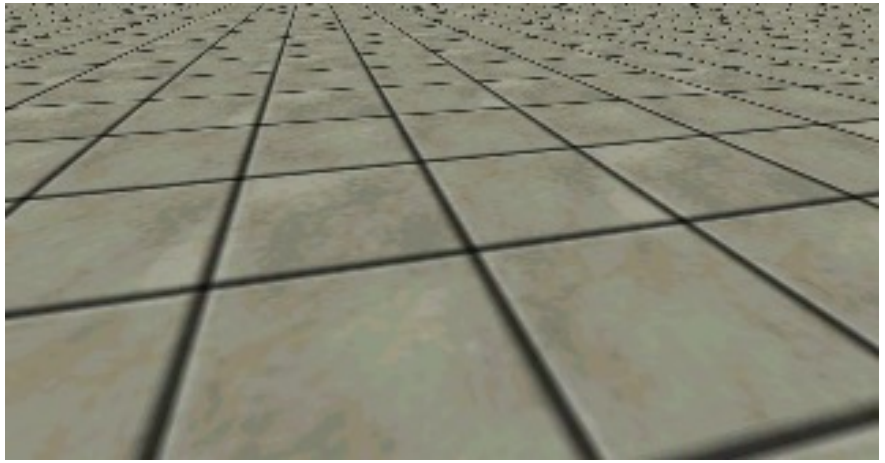
MIPMAPPING

Utiliser “une pyramide d’image” pour précalculer les versions moyennes d’image

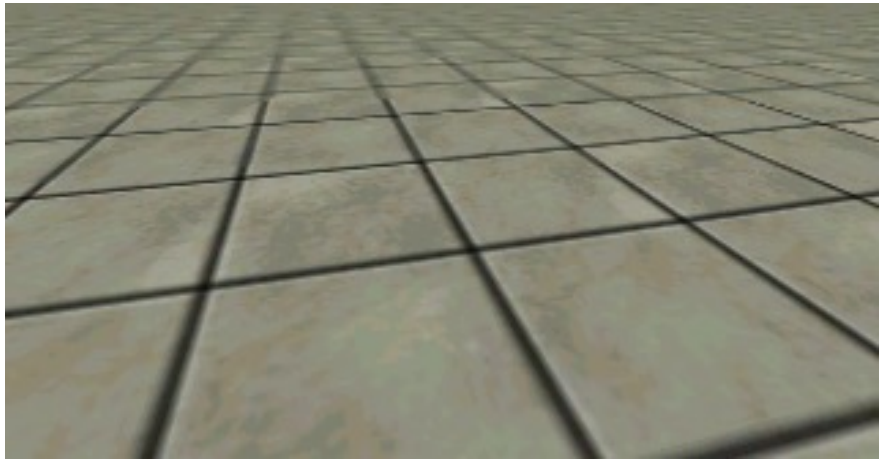


Stocker la pyramide entière comme un seul bloc de mémoire





Sans MIP-mapping

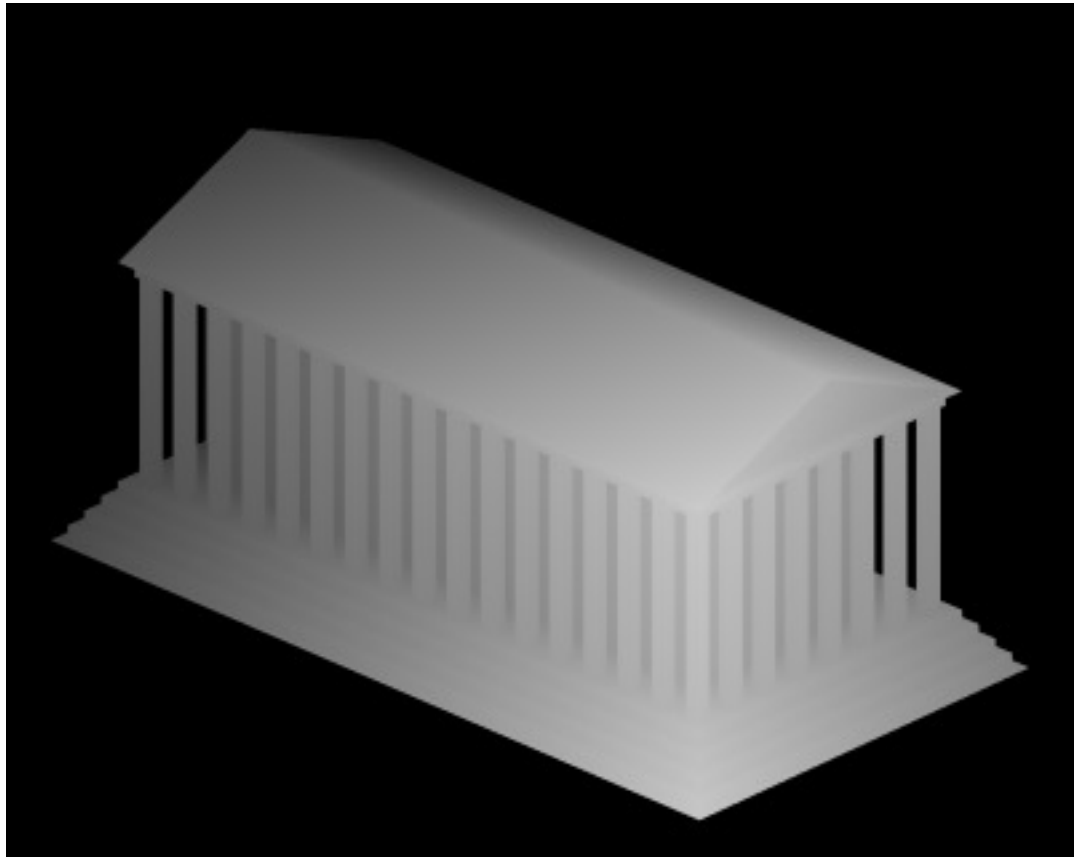


Avec MIP-mapping

LES OMBRES

On a besoin d'au moins 2 passes de *shaders* :

1. Rendre tout comme vu depuis **la lumière**
   Enregistrer la profondeur (tampon de profondeur, '*depth map*')



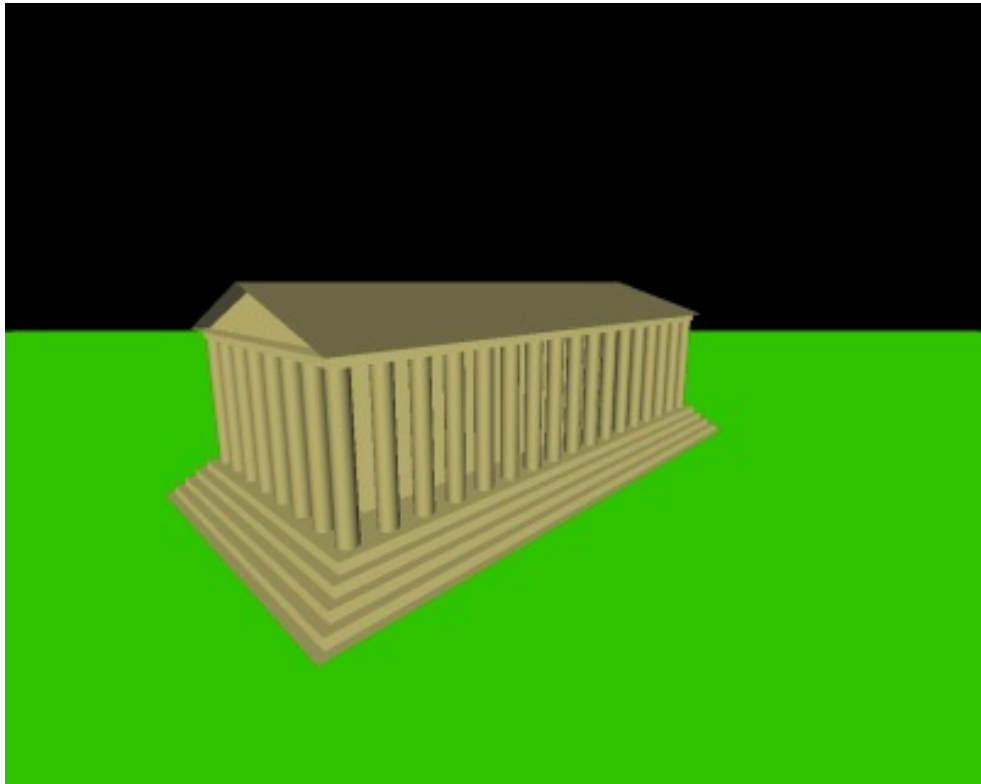
LES OMBRES (IDÉE)

On a besoin d'au moins 2 passes de *shaders* :

1. Rendre tout comme vu depuis **la lumière**
   Enregistrer la profondeur (tampon de profondeur/ombre, '*shadow map*')
2. Maintenant, rendre tout depuis la CAMÉRA
   Quand on calcule la couleur d'un fragment:

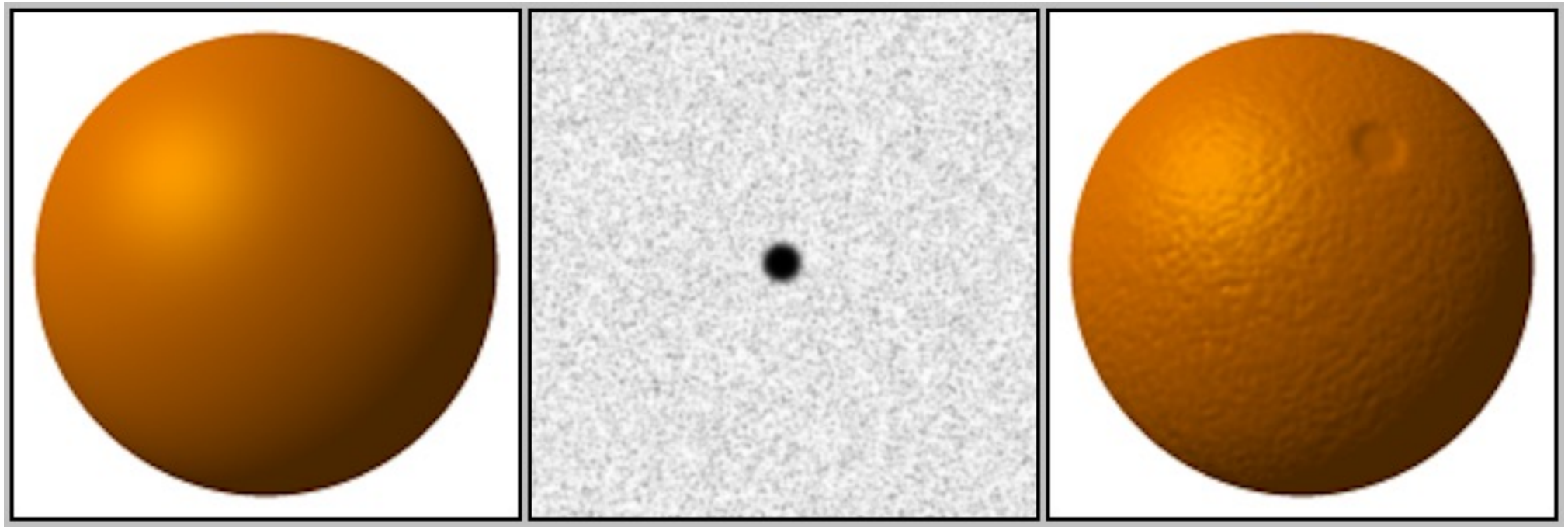
   • Convertir les coordonnées dans l'espace de la lumière (x_l, y_l, z_l)
      • Prendre la profondeur $D(x_l, y_l)$

   • Est-ce $z_l > D(x_l, y_l)$?
      • Oui: je suis dans l'ombre
      • Non: je suis éclairé



BUMP MAPPING ET NORMAL MAPPING

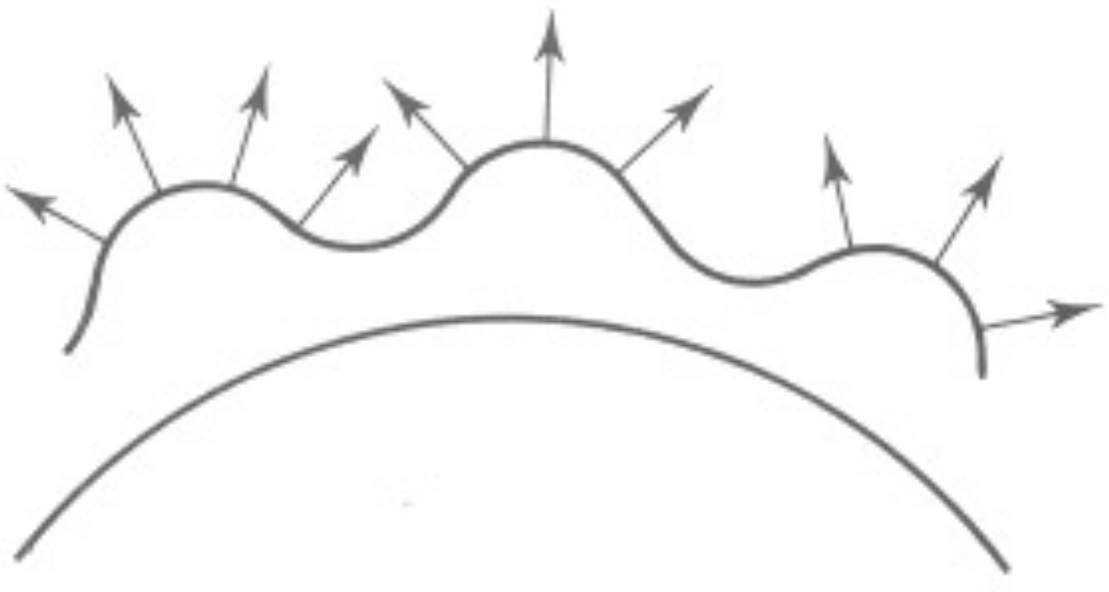
- La surface de l'objet est souvent rugueuse
  - On peut créer une géométrie plus complexe…
- Ou on peut la simuler en perturbant la normale seulement
  - Soit aléatoire
  - Soit spécifiée par une texture



BUMP MAPPING

$O'(u)$

Lengthening or shortening $O(u)$ using $B(u)$

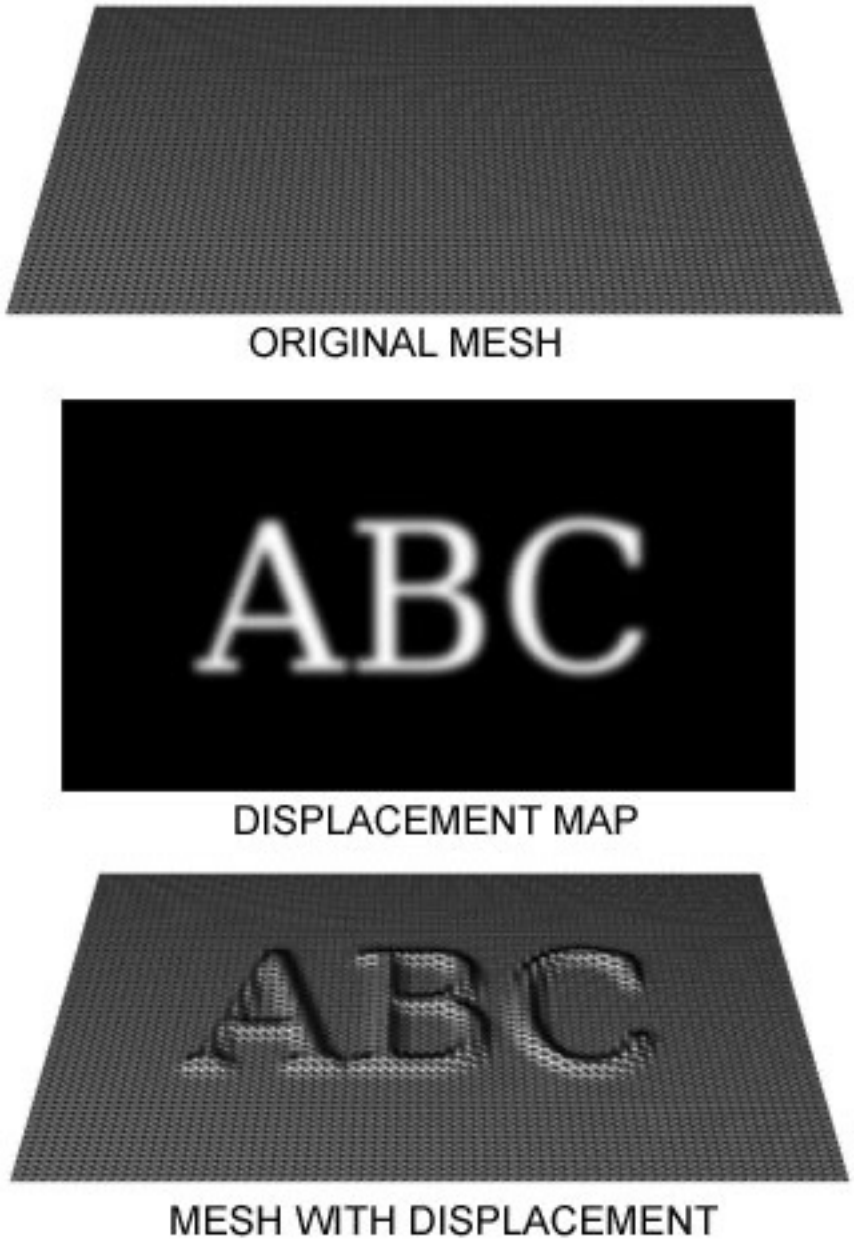


$N'(u)$

The vectors to the 'new' surface

DISPLACEMENT MAPPING

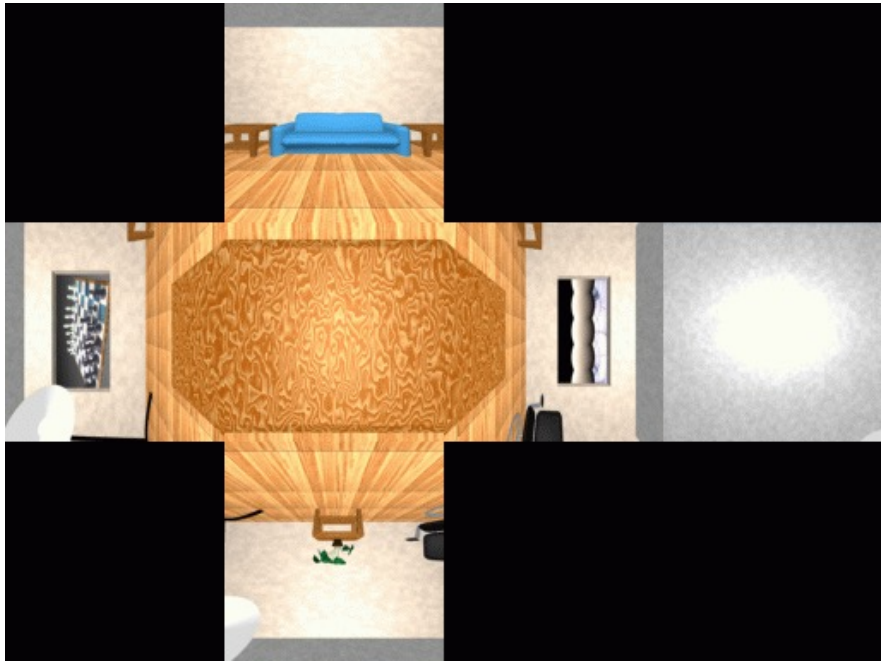
- Les silhouettes sont fausses après *le bump mapping*
  - Et les ombres aussi!
- Changer la géométrie en temps réel
  - On a besoin de subdiviser la surface



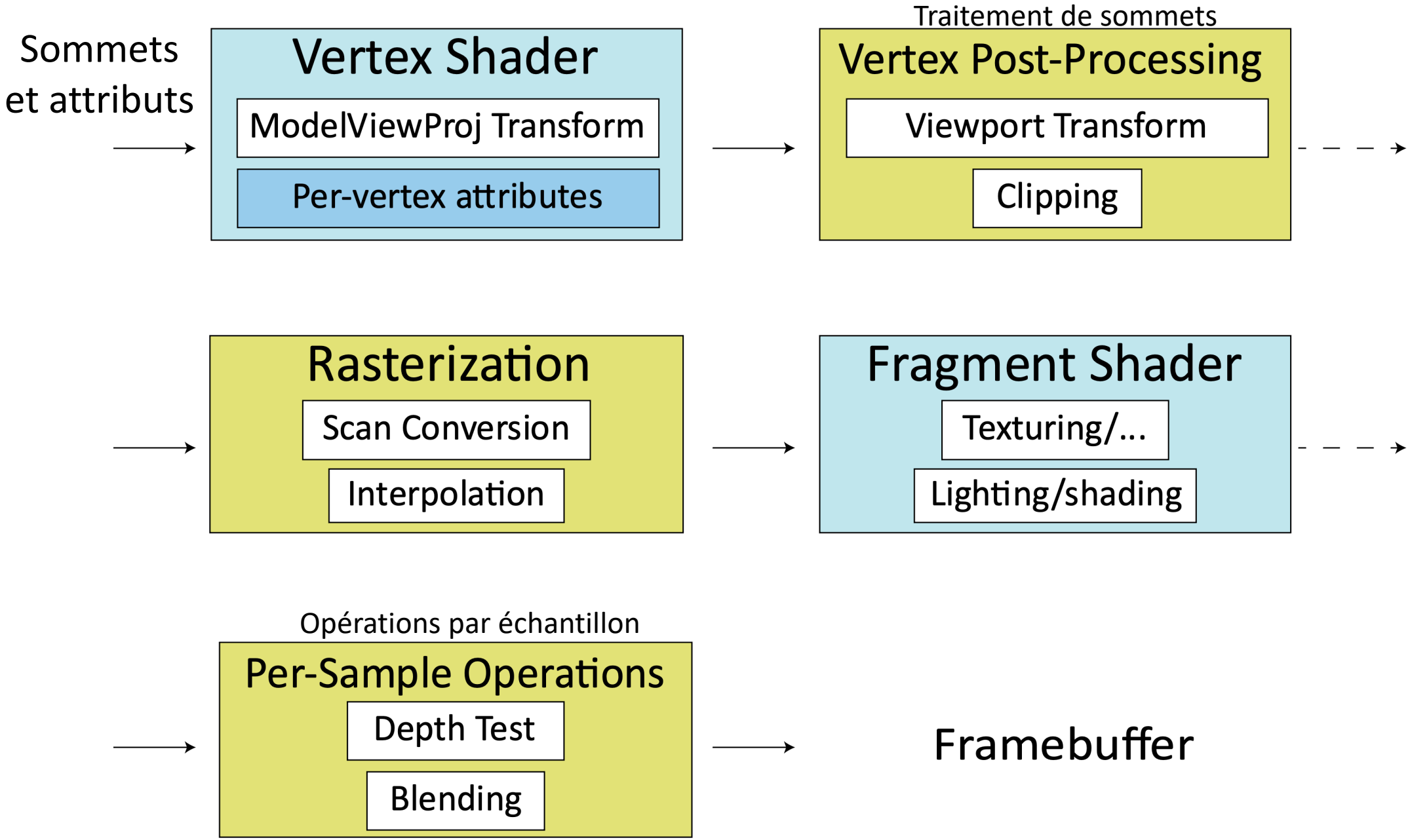
https://en.wikipedia.org/wiki/Displacement_mapping#/media/File:Displacement.jpg

CUBE MAPPING

- 6 textures planaires, les faces du cube
  - Pointe la caméra depuis l'origine dans 6 directions



PIPELINE: PLUS DE DÉTAILS



ALGORITHME DU PEINTRE: LES PROBLÈMES

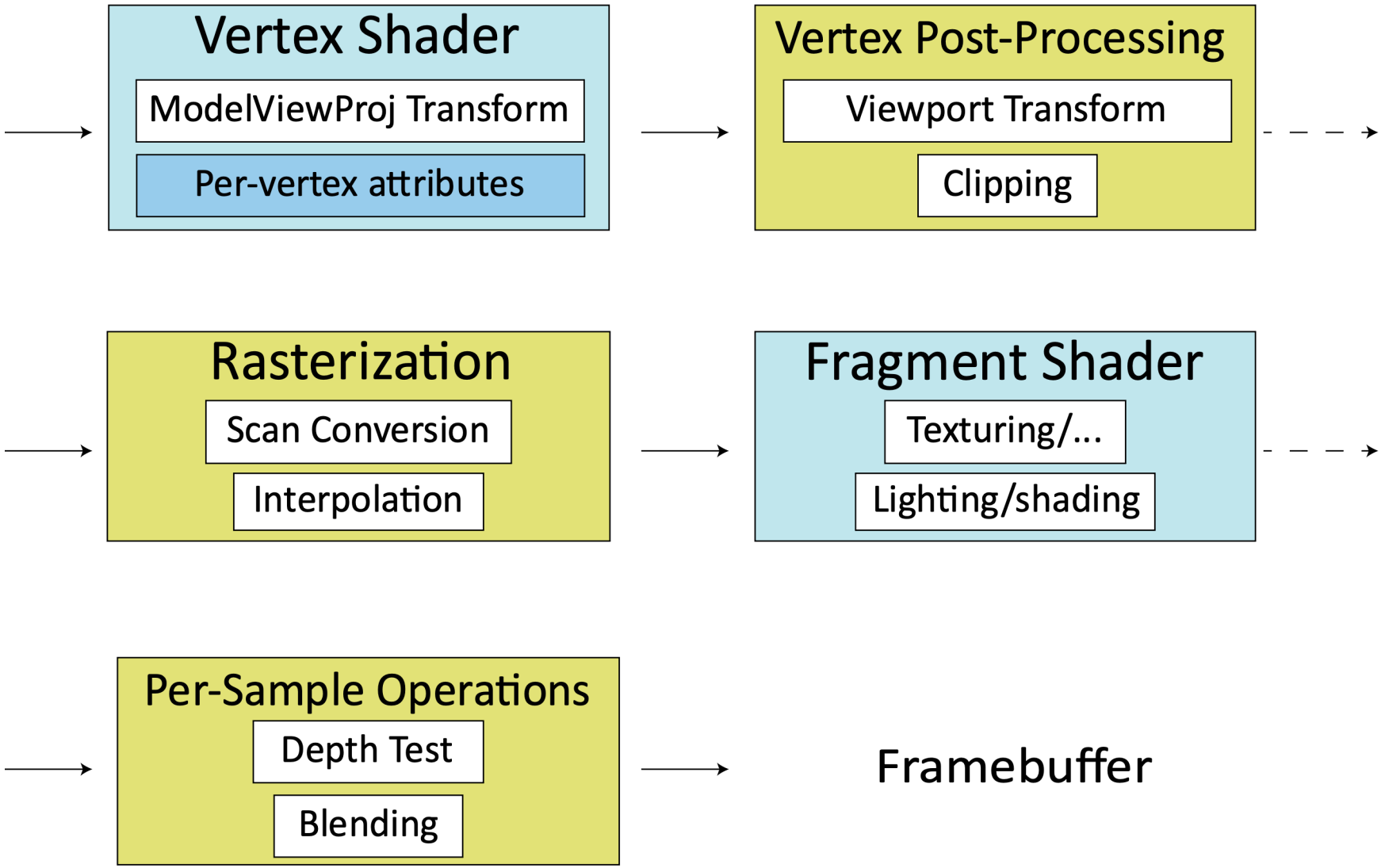
- Les polygones qui s'intersectent posent un problème
- Même les polygones sans intersections peuvent poser problème aussi:

TAMPON DE PROFONDEUR

- Stocker (r, g, b, z) pour chaque pixel
  - Normalement, 8+8+8+24 bits, peut-être plus

```
for all i,j {
 Depth[i,j] = MAX_DEPTH
 Image[i,j] = BACKGROUND_COLOUR
}
for all polygons P {
  for all pixels in P {
    if (Z_pixel < Depth[i,j]) {
      Image[i,j] = C_pixel
      Depth[i,j] = Z_pixel
    }
  }
}
```

POURQOUI APRÈS FRAGMENT SHADER??



LA PRÉCISION DU TEST DE PROFONDEUR

- Un rappel: une transformation projective transforme z du système de coordonnées de vue aux coordonnées normalisées (NDCS)
- L'exemple simple:

$$T\left(\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}\right) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & a & b \\ 0 & 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

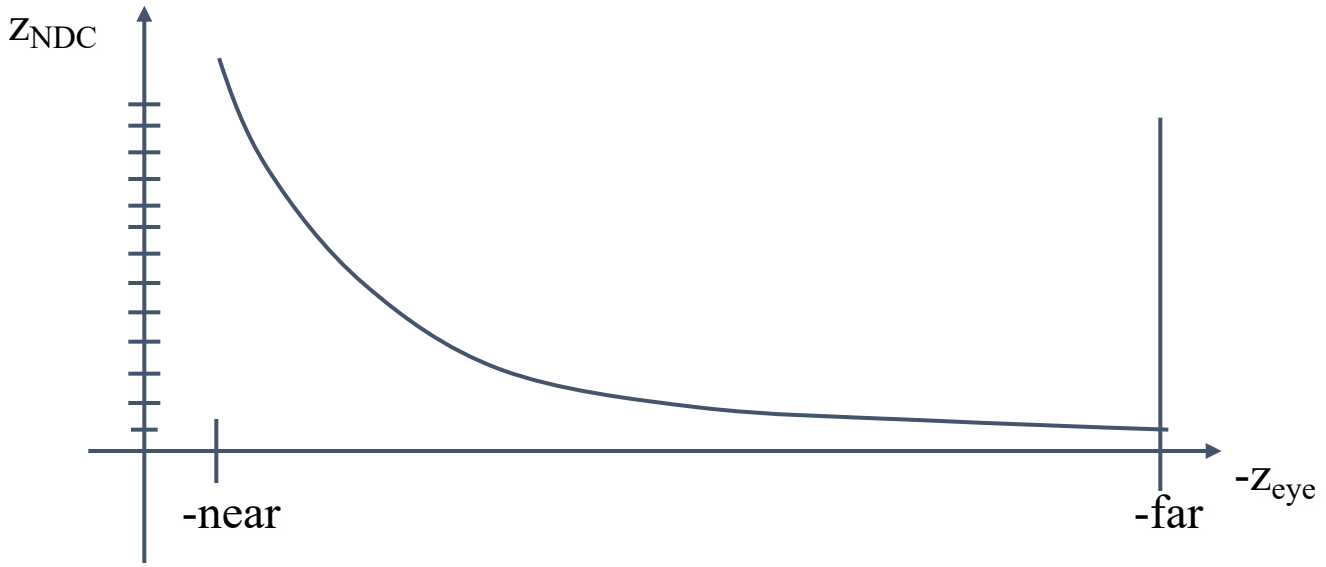
- Donc,

$$z_{NDC} = \frac{az_{eye} + b}{-z_{eye}} = -a - \frac{b}{z_{eye}}$$

LA PRÉCISION DU TEST DE PROFONDEUR

Donc, le tampon de profondeur stocke $1/z$ à la place de z!

- Les problèmes avec les tampons en entiers
  - Haute précision pour les objets proches
  - Faible précision pour les objets éloignés



LA PRÉCISION DU TEST DE PROFONDEUR

- La faible précision peut entraîner du Z-fighting pour les objets éloignés
  - Deux profondeurs différentes peuvent devenir la même valeur (quantization)
  - Quel objet gagne dépend de l'ordre de l'affichage

- Pire pour les grands ratios $\frac{f}{n}$
  - Règle générale: $\frac{f}{n} < 1000$ pour une profondeur sur 24 bits

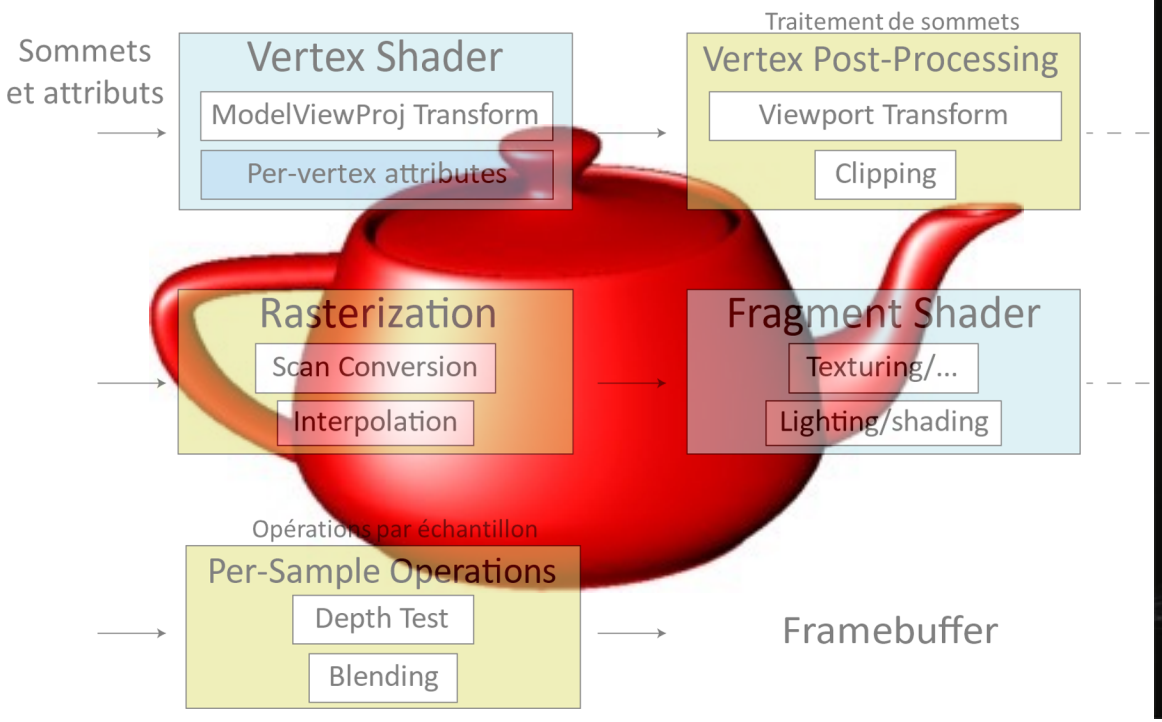
- Avec 16 bits on ne voit pas les différences de 1cm à la distance de 1km

LES MODÈLES ET ALGORITHMES D'ÉCLAIRAGE

L'éclairage local - rapide

- Plus loin de la physique
- Approximer l'apparence

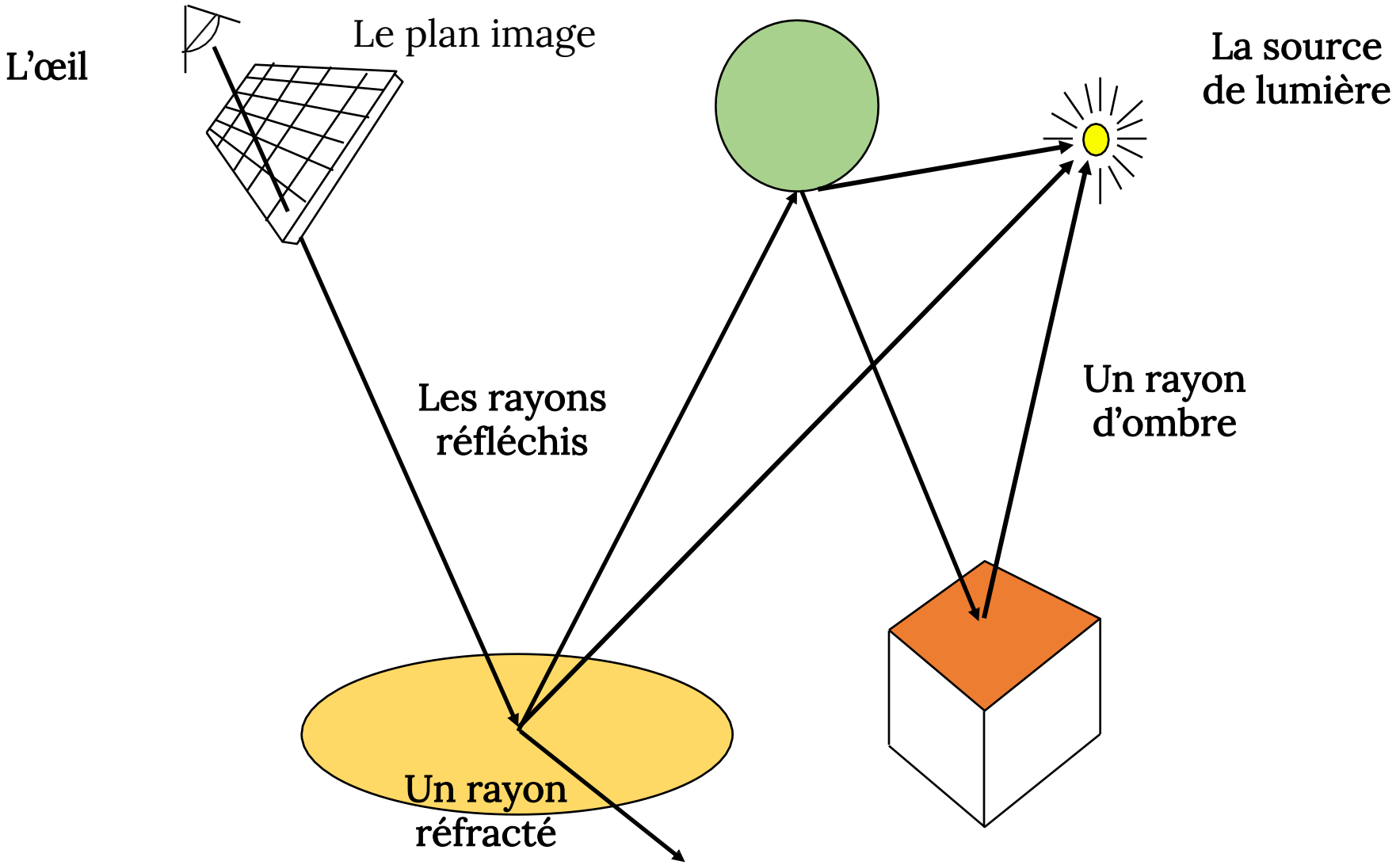
L'interaction de chaque objet avec la lumière



L'éclairage global – lent

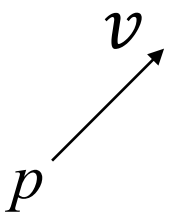
- Plus proche de la physique
- Les interactions entre les objets

L'IDÉE DU LANCER DE RAYONS



LES INTERSECTIONS

- La cœur du lancer de rayons ⇒ Elles doivent être calculées très rapidement
- Normalement, elles impliquent de résoudre des équations implicites



Exemple: L'intersection entre un rayon et une sphère

Un rayon: $x(t) = p_x + v_x t,\ y(t) = p_y + v_y t,$
$z(t) = p_z + v_z t$

Une sphère unitaire: $x^2 + y^2 + z^2 = 1$

L'équation quadratique en t:

$$0 = (p_x + v_x t)^2 + (p_y + v_y t)^2 + (p_z + v_z t)^2 - 1 =$$
$$= t^2(v_x^2 + v_y^2 + v_z^2) + 2t(p_x v_x + p_y v_y + p_z v_z)$$
$$+ (p_x^2 + p_y^2 + p_z^2) - 1$$

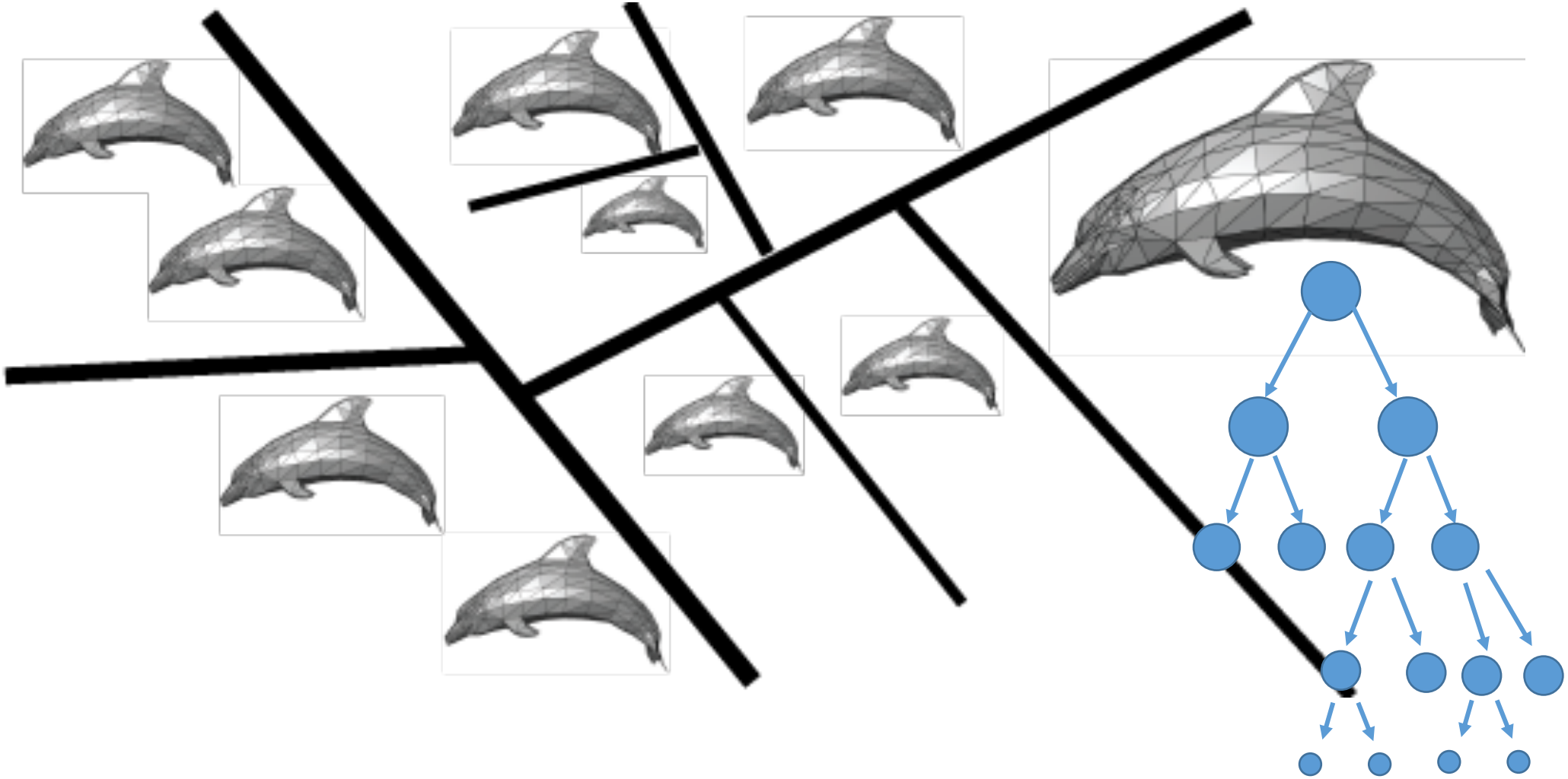
LANCER DE RAYONS: L'ÉCLAIRAGE DIRECT

Pour les maillages de triangles

- Interpoler l'information par sommets
  - Phong shading!
  - Comme qu'avant

- La différence entre le pipeline:
  - Il faut calculer les coordonnées barycentriques pour chaque point d'intersection

LES ARBRES BSP: UNE CONSTRUCTION



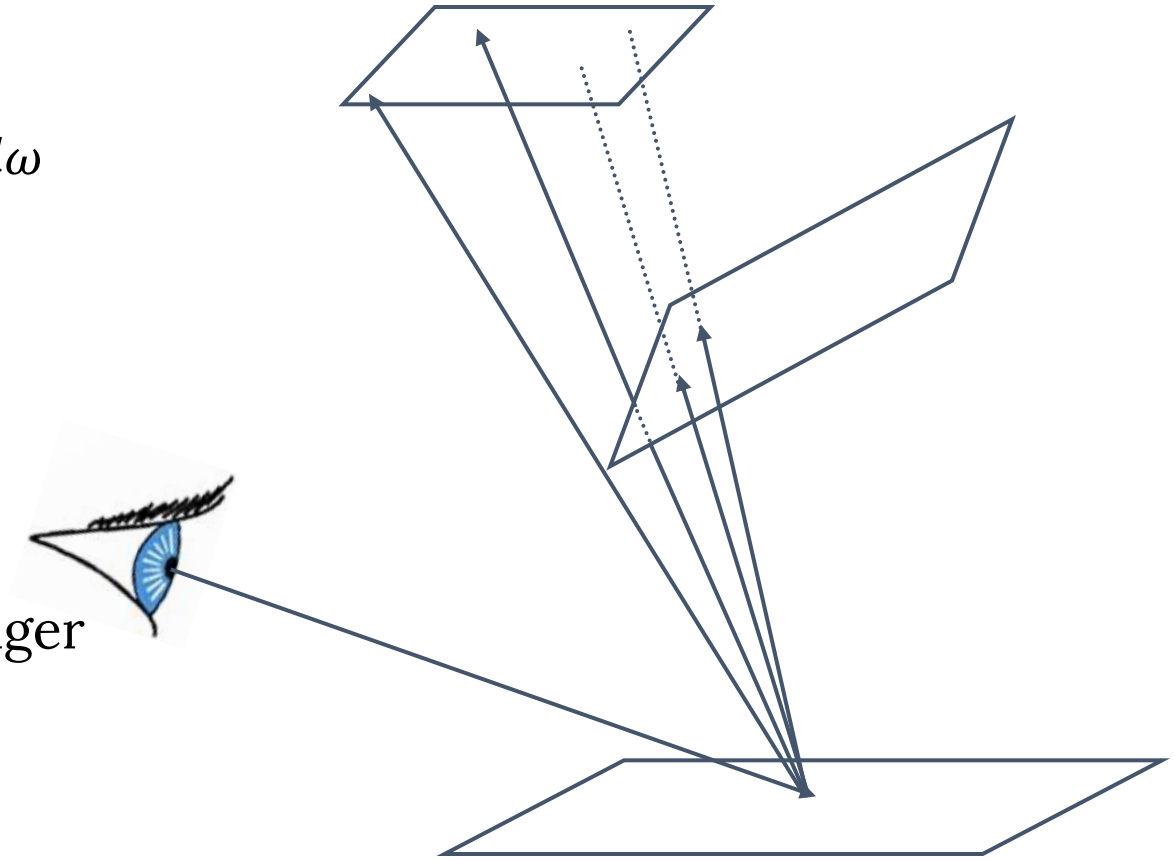
LES LUMIÈRES SURFACIQUES

- Les lumières surfaciques:
  - Nombre infini de rayons lumineux
  - Il faut intégrer:

$$I_{reflected} = \int_{\substack{light \\ directions}} \rho(\omega, \boldsymbol{n}) V(\omega) I_{light}(\omega) d\omega$$

- La visibilité et l'intensité peuvent changer pour différents points

Les lumières surfaciques



LES LUMIÈRES SURFACIQUES

Récrire l'intégrale, au lieu d'intégrer sur les directions

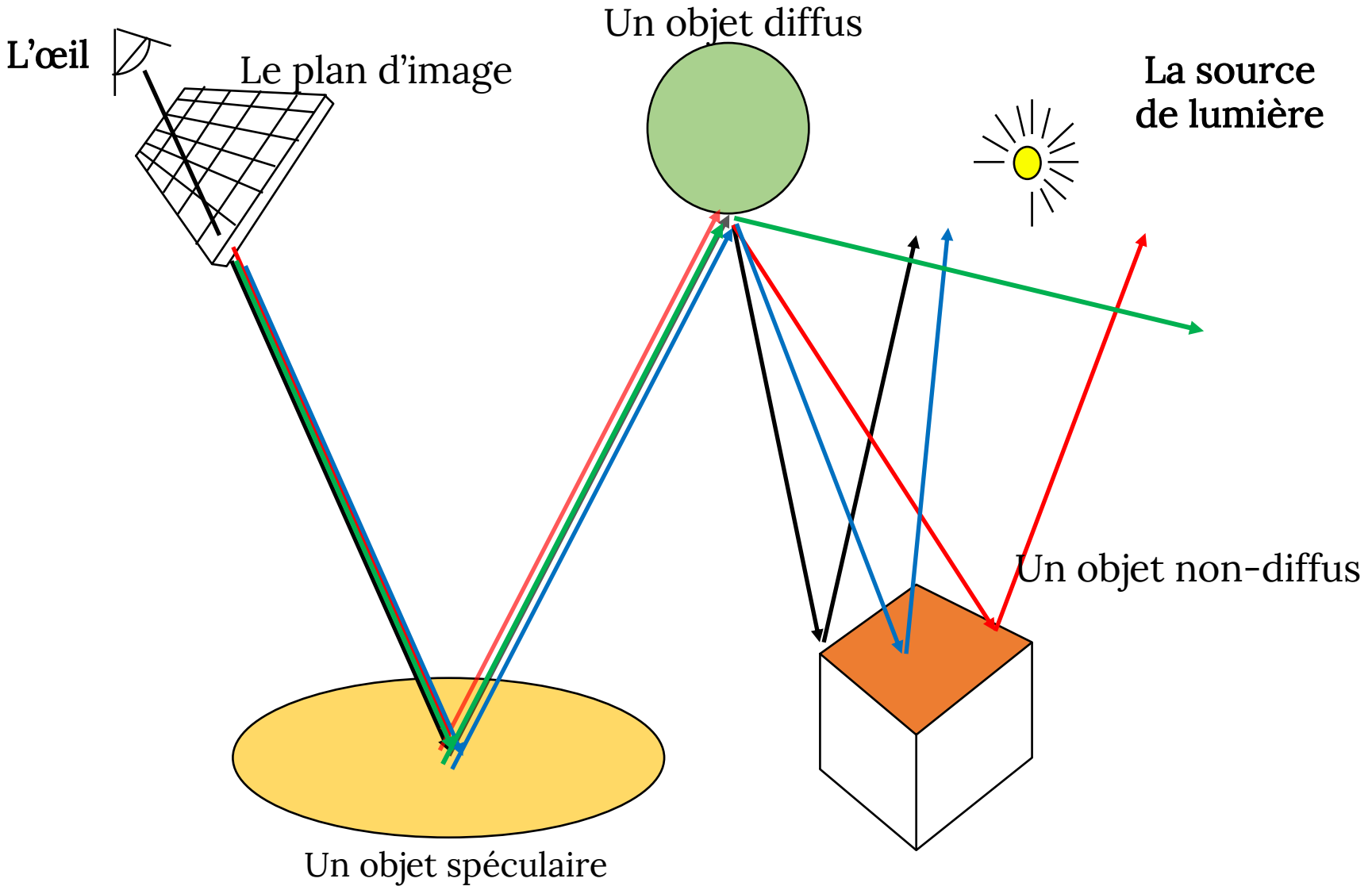
$$I_{reflected} = \int_{\substack{light \\ directions}} \rho(\omega, \boldsymbol{n}) V(\omega) I_{light}(\omega) d\omega$$

intégrer sur les points de la lumière

$$I_{reflected}(q) = \int_{s,t} \rho(p - q, \boldsymbol{n}) V(p - q) I_{light}(p) ds\, dt$$

- q est un point sur l'objet
- $p = F(s,t)$ est un point sur la lumière

TRACER DE CHEMINS



TRACER DE CHEMINS: LA VERSION LA PLUS SIMPLE

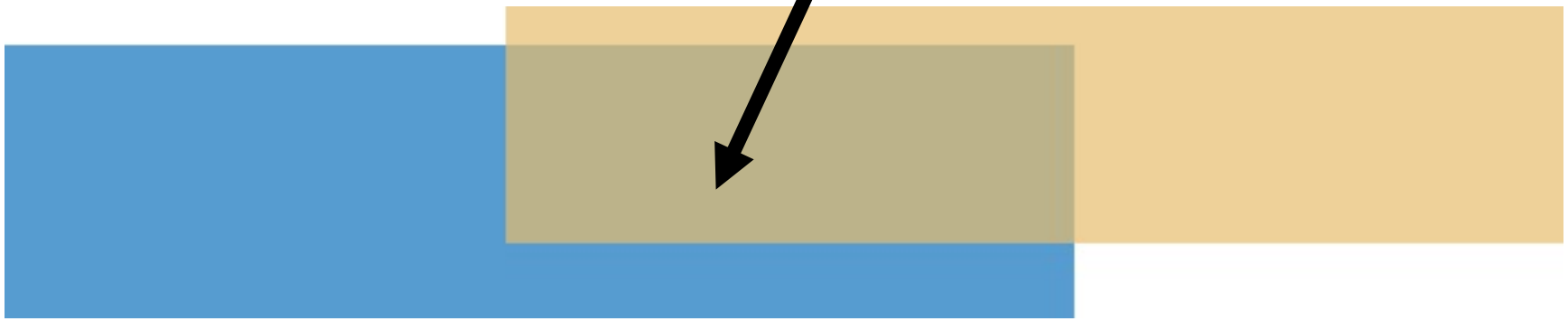
```
PathTrace(Ray r) {
P = closestIntersection(r);
if (random(emit, reflect) == emit)
return EmissionColor;
else {
Ray v = {intersectionPt,
randomDirectionInHemisphere(r.normalWhereObjWasHit)};
double cos_theta = dot(v.direction, r.normalWhereObjWasHit);
return PathTrace(v)*cos_theta*reflectance;
}
}
```

QUELS MÉLANGES PEUT-ON FAIRE?

- La transparence simple ("*over operator*"):
  - $f_1 = ADD, f_2 = ADD$
  - $d_1 = 1 - S.\alpha$
  - $s_1 = S.\alpha$
  - $d_2 = 0$
  - $s_2 = 1$

rgb: $(1 - 0.7) \cdot (\mathbf{0, 0, 1}) + 0.7 \cdot (\mathbf{1, 1, 0})$
= (0,0,0.3) + (0.7, 0.7,0)=**(0.7,0.7,0.3)**

$$Out.rgb = (1 - S.\alpha) \cdot D.rgb + S.\alpha \cdot S.rgb$$
$$Out.\alpha = 0 \cdot D.\alpha + 1 \cdot S.\alpha$$



$\boldsymbol{\alpha = 1.0}$ (opaque)

$\boldsymbol{\alpha = 0.7}$ (transparent)

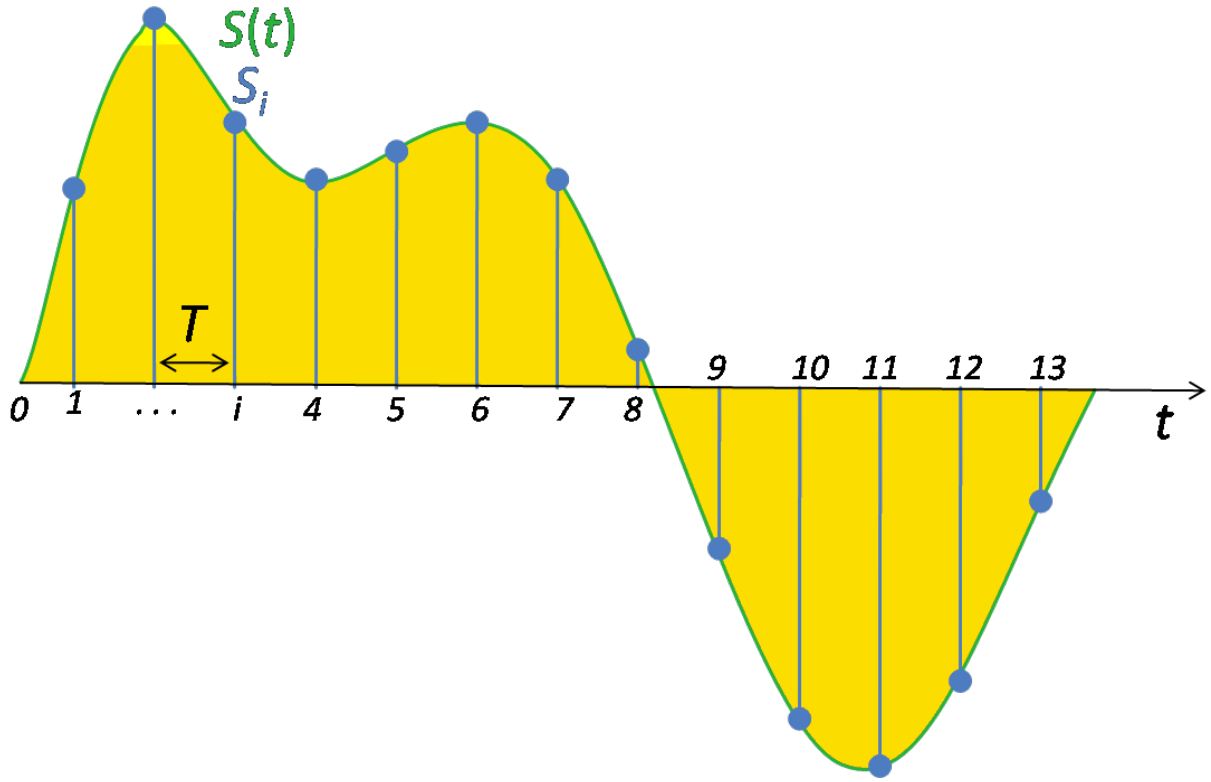
ALPHA BLENDING EN OPENGL

- *Alpha blending* est une opération dépendante de l'ordre!
  - Il importe quel objet est affiché en premier
  - Et quelle surface se trouve devant

- Pour les scènes 3D, il faut garder une trace de l'ordre de rendu: ***« Transparency Sorting »***
  - Afficher d'abord la surface arrière
  - Les surfaces opaques vont en premier
  - L'algorithme du peintre!
  - **On utilise le test de profondeur**
    - **e.g. pour afficher un objet opaque avant les objets translucides**

CONTINU VS DISCRET

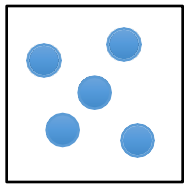
- Continu → Discret: ***Échantillonnage***
- Discret → Continu: ***Reconstruction/ Interpolation***



OVER-SAMPLING

- Même cette intégrale n'est pas facile à calculer
- Au lieu de cela, elle est approximée par une somme:

$$I[i,j] \leftarrow \frac{1}{n}\sum_{k=1}^{n} I(x_k, y_k)$$



Où k indexe un ensemble de points (x_k, y_k) appelés les positions d'échantillons

Il faut rendre la version de haute résolution pour la couleur est la profondeur

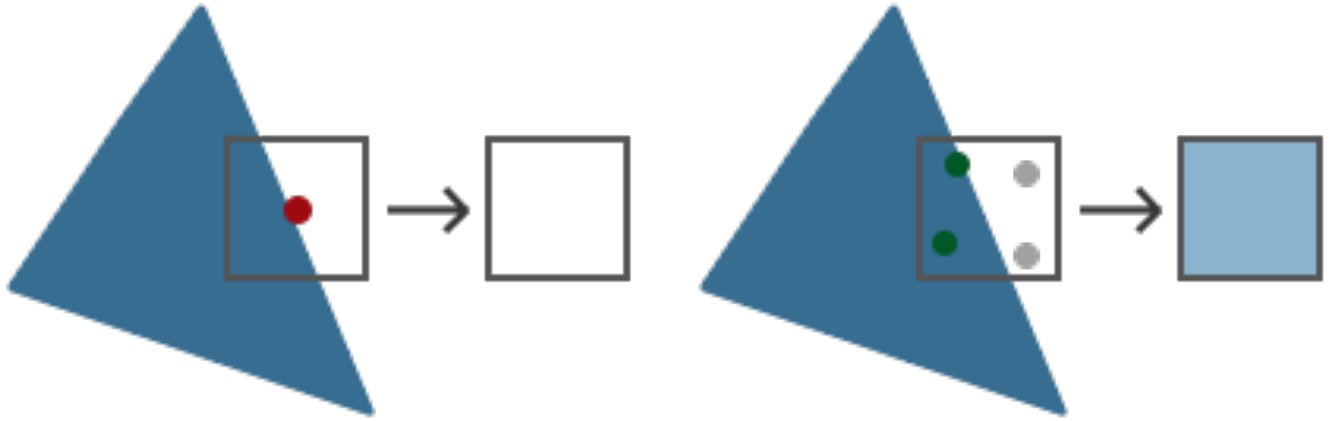
- On utilise le mot «échantillon» pour les pixels des images de hautes résolutions

SUPER-SAMPLING

- Si les position d'échantillonnage sont sur grille régulière, c'est *super-sampling.*
- On peut utiliser les autres modèles des positions
  - Les modèles moins réguliers peuvent éviter les erreurs systématiques (biais)

MULTI-SAMPLING

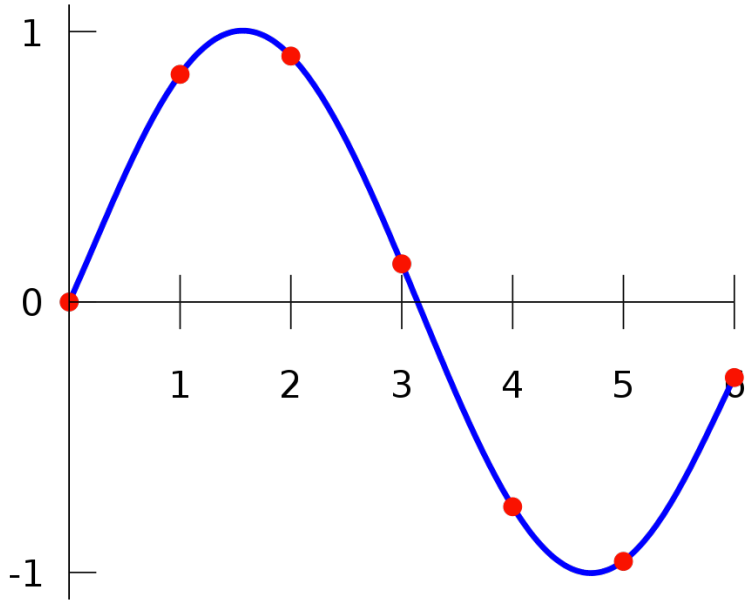
- Faire le rendu en haute résolution (la couleur + profondeur)
- *Pendant la rastérisation* de chaque triangle, “la couverture” et les valeurs *z* sont calculées à ce niveau
- (efficacité) Mais le *fragment shader* est exécuté **une fois par pixel final (par triangle)**



https://learnopengl.com/Advanced-OpenGL/Anti-Aliasing

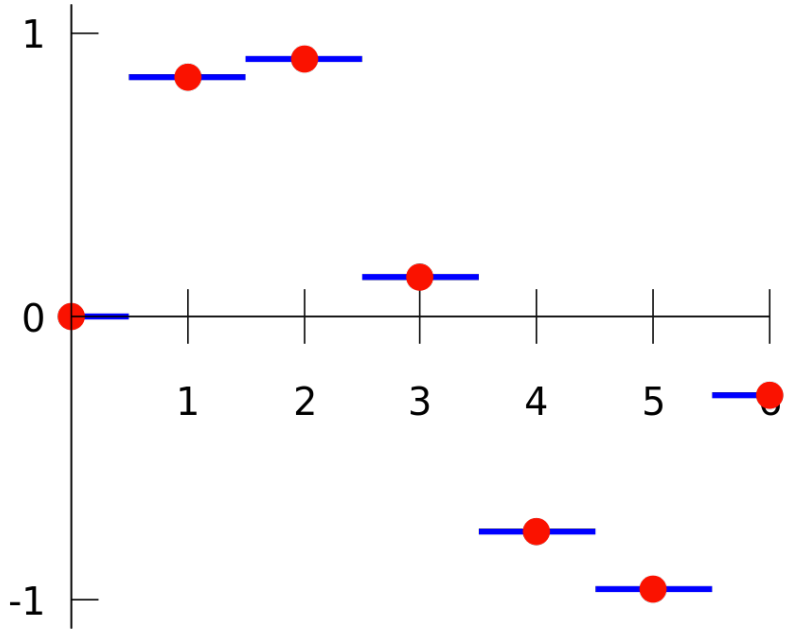
QU'EST-CE QUE L'INTERPOLATION?

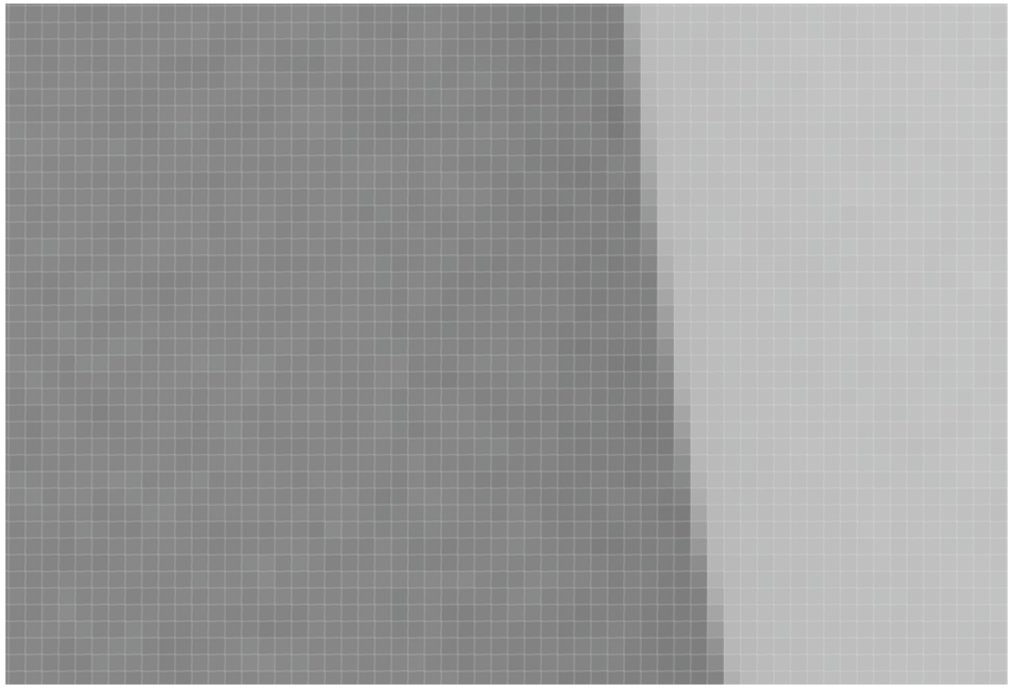
- Intuitivement: remplir les valeurs entres celles qu'on connait
- Plus formellement:
  - Soit $\{x_i, y_i\}, i = 1 \dots N$
  - Trouver une fonction $f(x)$, telle que $\boldsymbol{f(x_i) = y_i}, i = 1 \dots N$



L'INTERPOLATION CONSTANTE

- Aussi appelée « le plus proche voisin»
- Très simple
- Mais discontinue

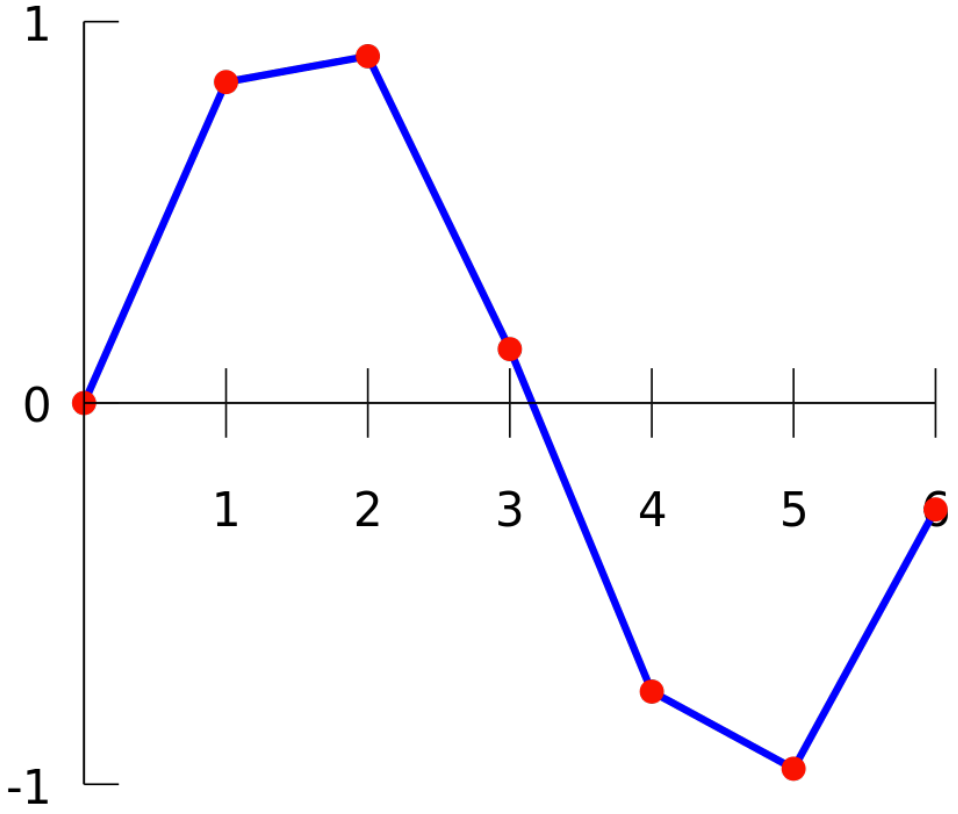




L'INTERPOLATION LINÉAIRE

- Dessiner une ligne entre chaque paire de points consécutifs

- $$f(x) = \begin{cases} a_1 x + b_1, if\ x_0 \leq x < x_1 \\ a_2 x + b_2, if\ x_1 \leq x < x_2 \\ a_3 x + b_3, if\ x_2 \leq x < x_3 \\ \dots \end{cases}$$



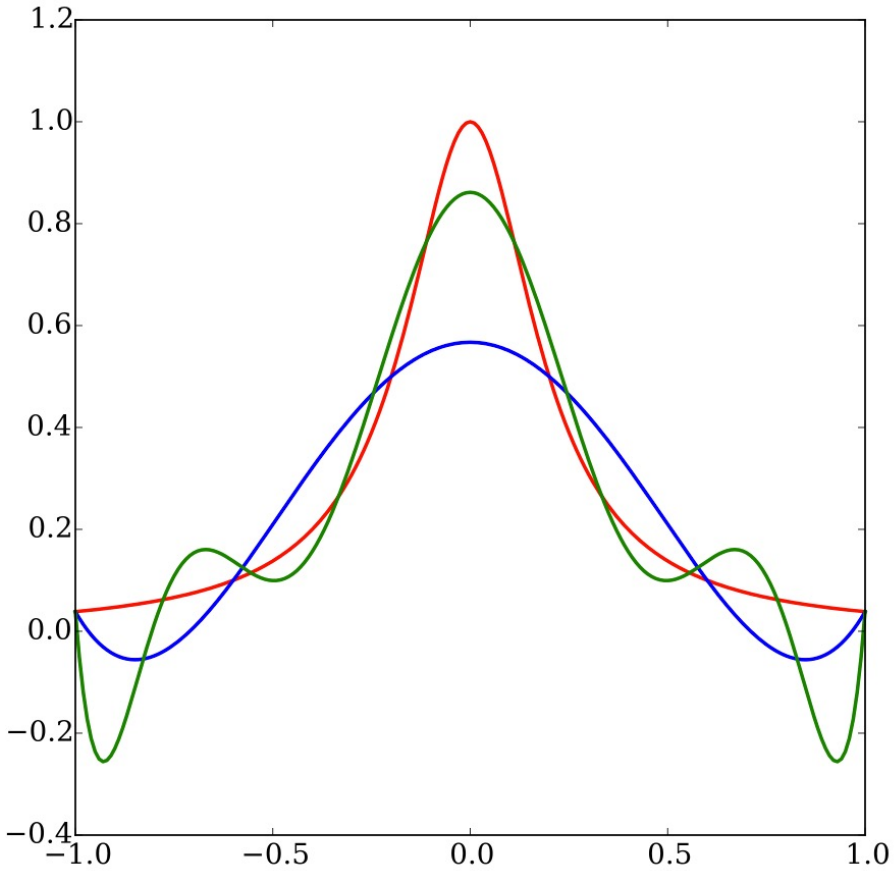
L'INTERPOLATION POLYNOMIALE

- Peut-on trouver une seule fonction qui passe par ces points?
- Oui:
- Pour *n+1* points:
  - $f(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n$

- Comment trouver a_i?
  - $f(x_i) = y_i$
  - $n + 1$ points, $n + 1$ inconnues

PHÉNOMÈNE DE RUNGE

Si le degré du polynôme est plus élevé, la fonction devient ondulée



La fonction qu'on interpole

Le polynôme de degré 5

Le polynôme de degré 9

LES SPLINES

- On limite le degré du polynôme à quelque chose petit, e.g. 3
- Utiliser les fonctions par morceaux
- On peut rendre la fonction lisse

$$f_x(x) = \begin{cases} \frac{1}{4}(x+2)^3 & -2 \leq x \leq -1 \\ \frac{1}{4}\left(3|x|^3 - 6x^2 + 4\right) & -1 \leq x \leq 1 \\ \frac{1}{4}(2-x)^3 & 1 \leq x \leq 2 \end{cases}$$

LA BASE CUBIQUE HERMITE

- Les coefficients ont une signification géométrique
  - 2 positions + 2 tangentes
- Les exigences: $\begin{cases} C(0) = P_0 \\ C(1) = P_1 \\ C'(0) = T_0 \\ C'(1) = T_1 \end{cases}$

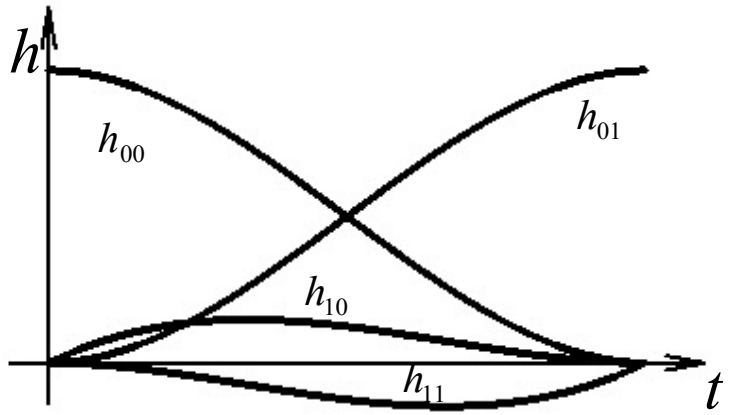
- Définir une fonction de base par contrainte:

$$C(t) = P_0 h_{00}(t) + P_1 h_{01}(t) + T_0 h_{10}(t) + T_1 h_{11}(t)$$

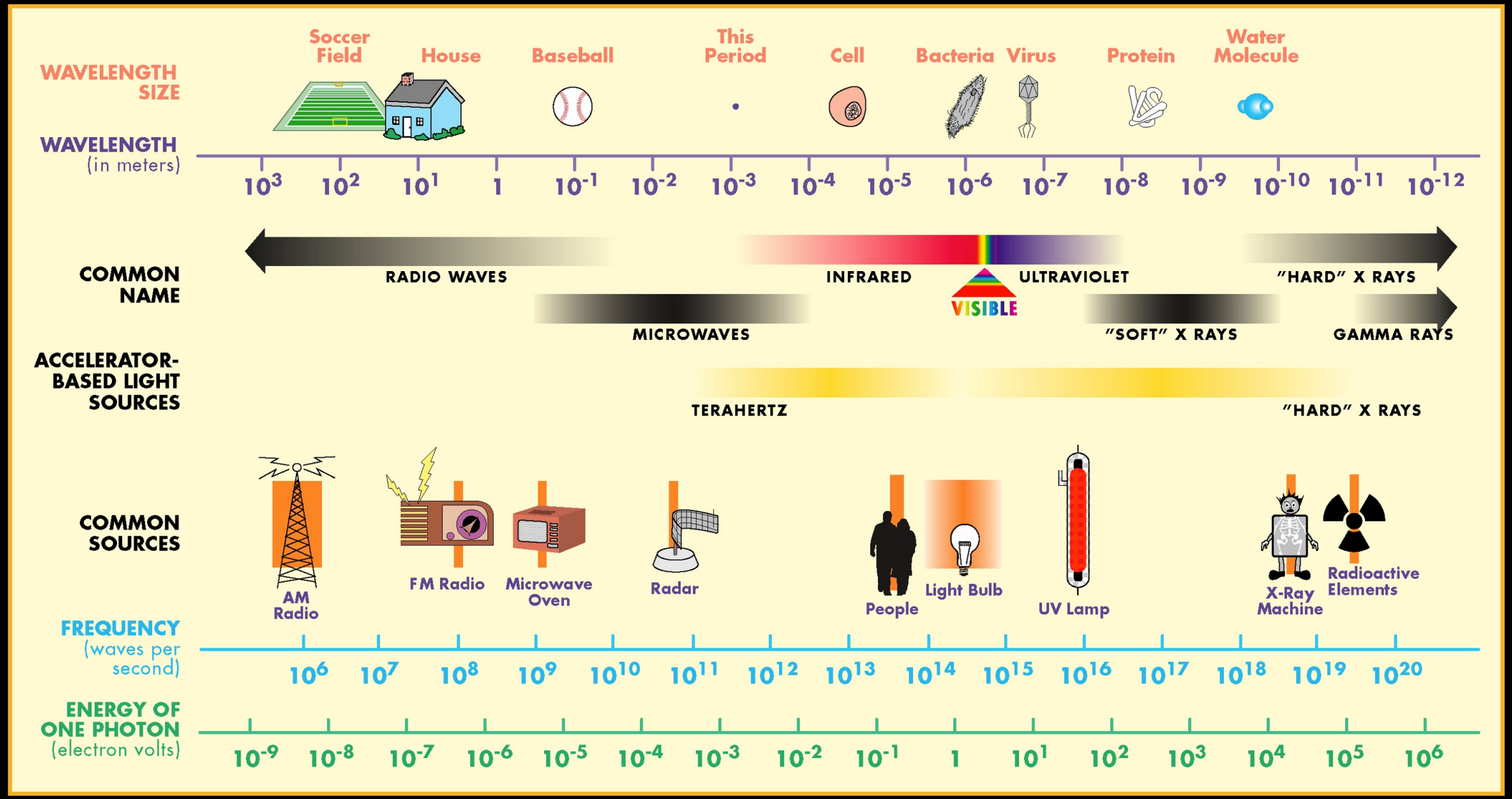
LA BASE CUBIQUE HERMITE

Les quatre polynômes qui satisfont les contraintes

$$h_{00}(t) = t^2(2t-3)+1 \qquad h_{01}(t) = -t^2(2t-3)$$
$$h_{10}(t) = t(t-1)^2 \qquad h_{11}(t) = t^2(t-1)$$

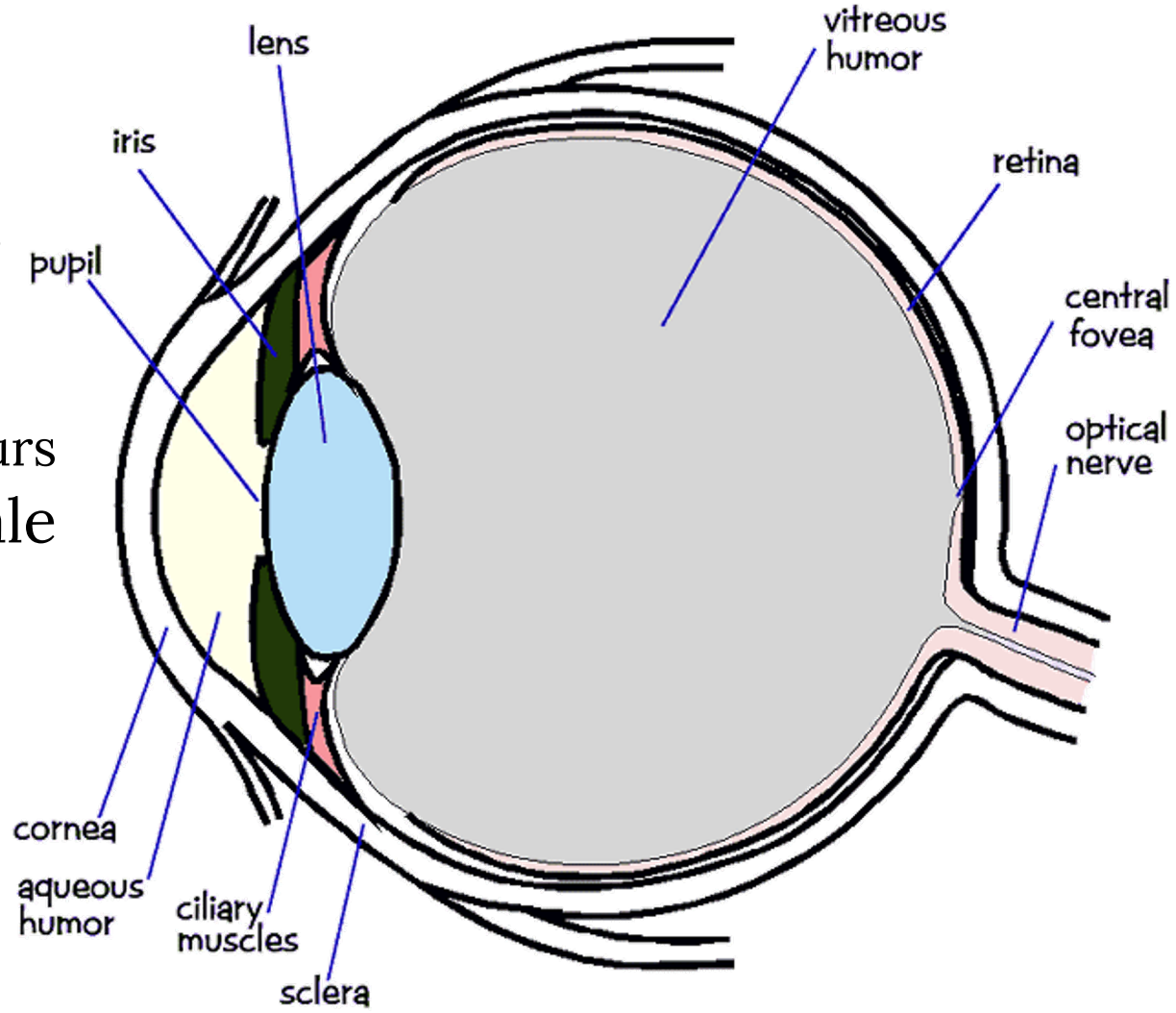


THE ELECTROMAGNETIC SPECTRUM



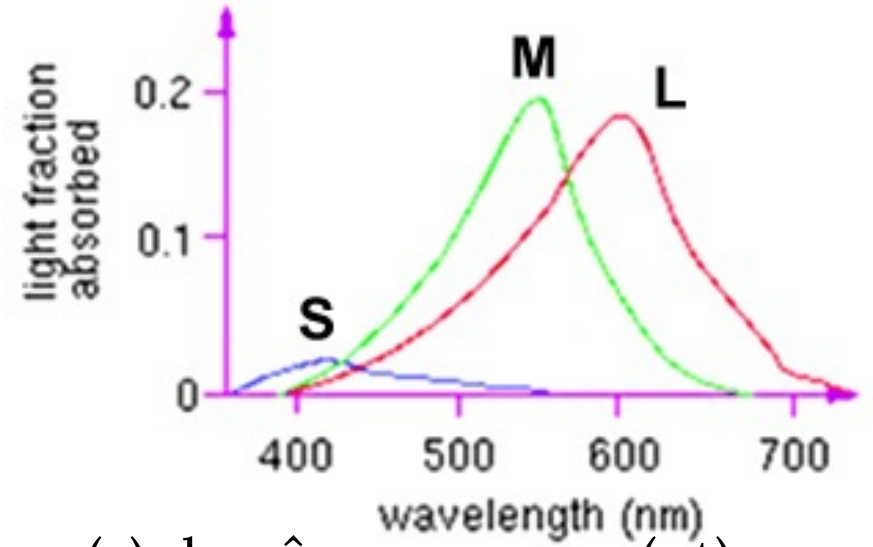
LA PHYSIOLOGIE DE LA VISION

- La rétine
  - Les bâtonnets
    - Nuances de gris, bords
  - Les cônes
    - 3 types
    - Les senseurs de couleurs
  - Une distribution inégale
    - La fovéa est dense



LA TRICHROMIE

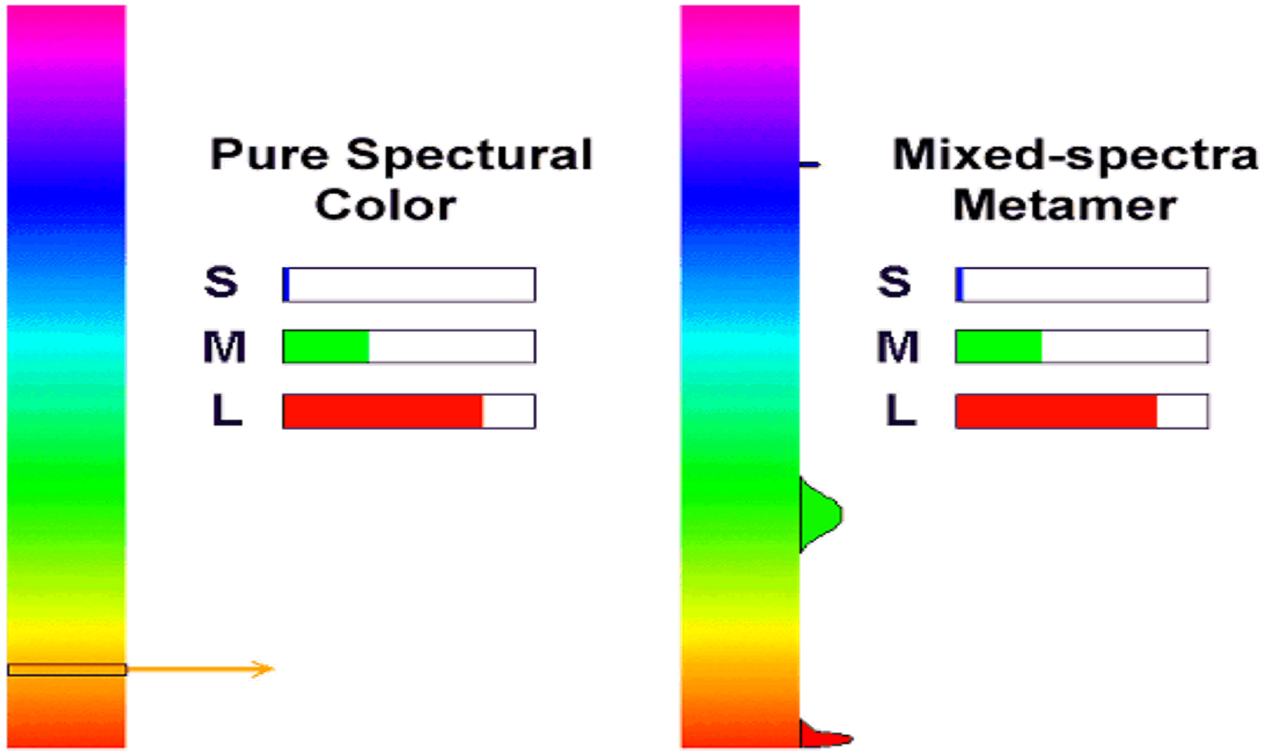
- Trois types de cônes
  - L ou R, plus sensible à la lumière rouge (610 nm)
  - M ou G, plus sensible à la lumière verte (560 nm)
  - S ou B, plus sensible à la lumière bleue (430 nm)



  - Le daltonisme = type(s) de cônes manque(nt)

METAMERS

- La couleur qu'on voit dérive d'une combinaison de stimuli de chaque type de cônes

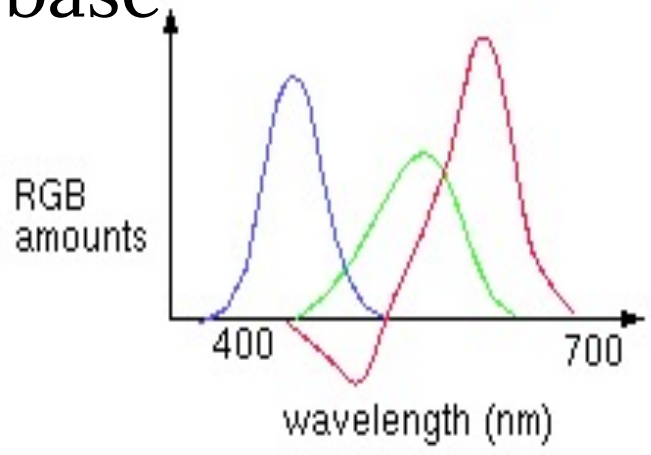


- La même couleur peut être causée par des spectres différents
- démo

http://www.cs.brown.edu/exploratories/freeSoftware/catalogs/color_theory.html

LOBES NÉGATIFS

- Parfois les trois composants ne pouvaient pas correspondre parfaitement!
  - Souvent RVB a produit « trop de rouge »
  - Mais ce n'est pas possible de retirer le rouge d'un écran
- Donc, on ne peut pas générer toutes les couleurs avec n'importe quelles couleurs de base (primaires)
- La solution: choisir de nouvelles couleurs de base (synthétiques)



L'ESPACE DE COULEUR CIE

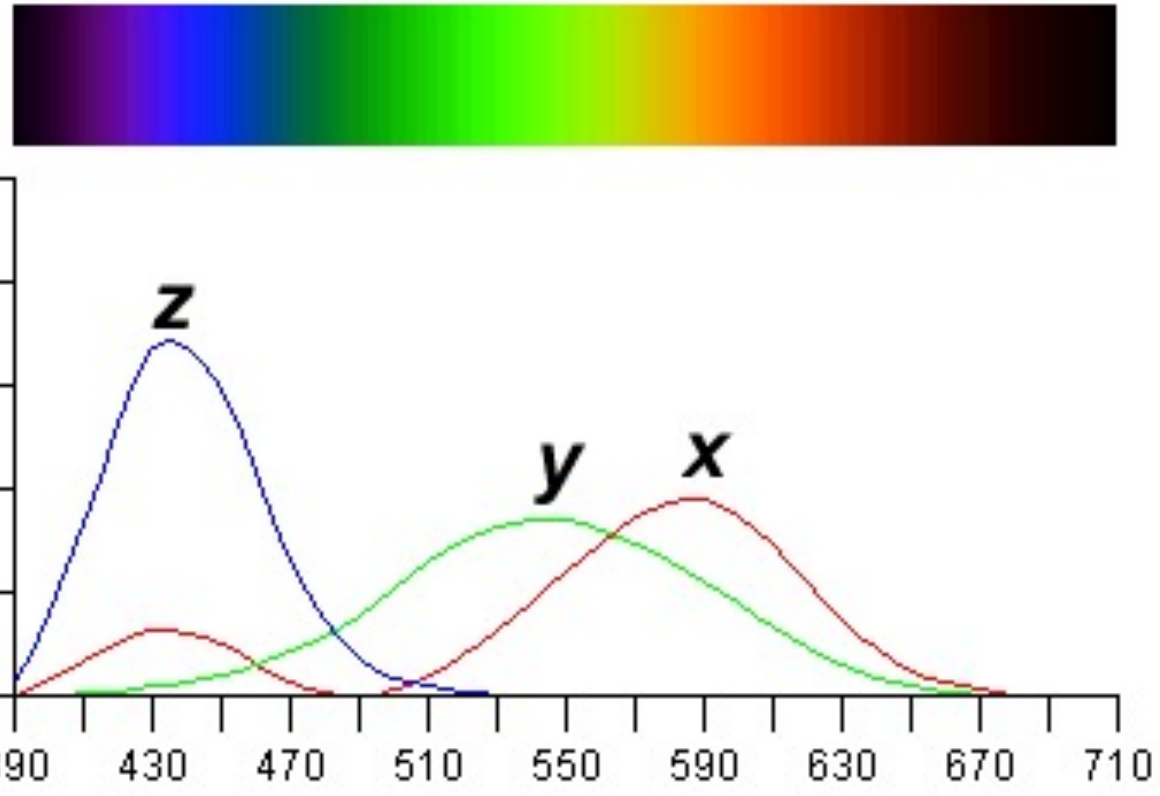
- CIE a défini 3 lumières X, Y, Z imaginaires
  - Toutes les longueurs d'onde λ peuvent être construites par une combinaison positive
  - Les fonctions de base!

NB:

$X \sim R$

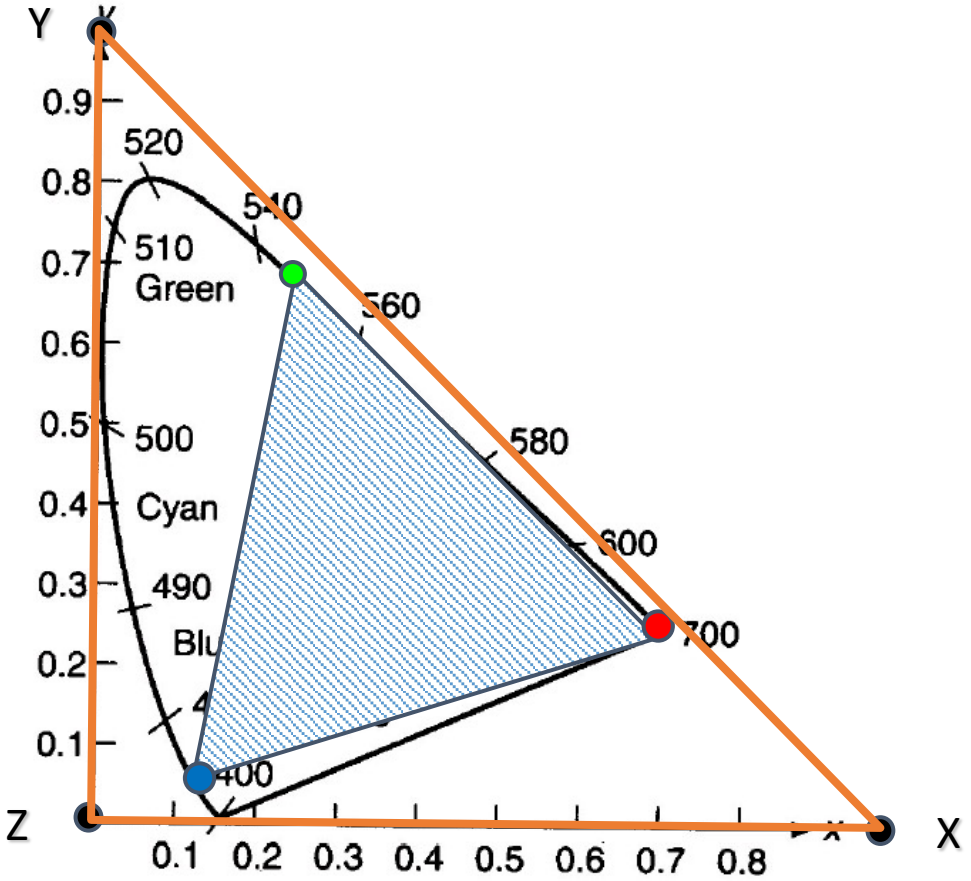
$Y \sim G$

$Z \sim B$



RGB VS XYZ UNE AUTRE FOIS

Une autre explication pourquoi la courbe R parfois doit être négative



L'INTERPOLATION
LES COULEURS COMPLÉMENTAIRES
LA LONGUEUR D'ONDE DOMINANTE